VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Použití dopravního simulátoru MATSIM

Using Transport Simulator MATSIM

2017 Tomáš Minařík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student:	**Bc. Tomáš Minařík**
Studijní program:	N2647 Informační a komunikační technologie
Studijní obor:	2612T025 Informatika a výpočetní technika
Téma:	Použití dopravního simulátoru MATSIM Using Transport Simulator MATSIM
Jazyk vypracování:	čeština

Zásady pro vypracování:

V rámci diplomové práce bude student pracovat s open source nástrojem MATSIM pro multi-agentní dopravní simulace. Cílem práce je prostudovat možnosti tohoto software v oblasti implementace nových modulů pro přidávání externích dat a také pro úpravu algoritmů. Toto bude vyžadovat nastudování problematiky multi-agentních simulací a následně detailně projít objektový návrh simulátoru MATSIM. Výstupem budou následně ukázkové implementace vybraných modulů do simulátoru MATSIM a provedení experimentů na testovacích datech.

Jednotlivé cíle práce jsou:
1. Nastudování problematiky multi-agentních dopravních simulací.
2. Prostudovat software MATSIM a spustit v něm experimentální simulaci.
3. Prostudovat možnosti implementace vlastních rozšíření do software MATSIM.
4. Implementace speciálních rozšíření do softwre MATSIM a provedení experimentů.

Seznam doporučené odborné literatury:

[1] MATSIM: Agent-Based Transport Simulations, http://www.matsim.org

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Olomouci 28. dubna 2016

Minich Tomáš

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 28. duben 2017

Mimich Tomáš .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli. Především vedoucímu diplomové práce Ing. Janu Martinovičovi, Ph.D za jeho cenné rady a přípomínky. Také rodině, protože bez ní by tato práce nevznikla.

Abstrakt

Diplomová práce se zabývá open source nástrojem MATSim - multiagentním dopravním simulátorem. Na úvod jsme v práci probrali pojem modelování dopravy a pojem agent. Díky těmto znalostem jsme si mohli definovat pojem multiagentní simulace. Po představení teoretické části práce jsme zmíněné pojmy zasadili do prostředí MATSimu. MATSim byl podrobně představen z teoretického hlediska. Bylo ukázáno, jak iterativně ohodnocuje simulaci, jaké důležité soubory pro simulaci potřebuje a jak je nakonfigurovat. Představili jsme detailní implementace specifických problémů v MATSimu. Výsledkem práce jsou dvě naimplementovaná rozšíření. První rozšíření se týká přeplánování tras agenta během dne, druhé ukazuje možnost ovlivnění výběru trasy pomocí dijkstrova algoritmu. V poslední části byly provedeny pokusy. Bylo zjištěno chování ohodnocovací funkce dle počtu iterací, rychlost adaptace na lepší řešení a zda se zkrátí čas výpočtu použitím paralelizace. Po prostudování práce by měl být čtenář schopen napsat vlastní rozšíření do MATSimu.

Klíčová slova: MATsim, modelování dopravy, simulace dopravy, diplomová práce, Java

Abstract

The diploma thesis deals with open source tool MATSim - multiagent traffic simulator. At the beginning, we discussed the concept of transport modeling and the concept of agent. Thanks to this knowledge, we could define the concept of multiagent simulation. After introducing the theoretical part of the thesis we put these concepts into the MATSimu environment. MATSim was presented in detail from a theoretical point of view. It has been shown how iteratively evaluates the simulation, what important simulation files it needs and how to configure it. We introduced a detailed implementation of specific problems in MATSim. Result of the work is two implemented extensions. The first extension involves re-planning the route of the agent during the day, the second showing the possibility of influencing route selection using the dijkstr algorithm. In the last part tests were made. Behavior of the evaluation function was determined by the number of iterations, the rate of adaptation to a better solution, and shortening calculation time by using parallelization. After reading the work, reader should be able to write his own extension to MATSim.

Key Words: MATSim, traffic modeling, traffic simulation , master thesis, Java

Obsah

Seznam použitých zkratek a symbolů

MATSim	–	Multi-Agent Transport Simulation
xml	–	eXtensible Markup Language
UTM	–	Univerzální transverzální Mercatorův systém souřadnic
VTTS	–	value of travel time savings
SDK	–	Software development kit
IDE	–	Integrated Development Environment
png	–	Portable Network Graphics

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V reálném světě jsou některé problémy příliš složité pro studium a pochopení. Jejich studium by vyžadovalo velké prostředky, nebo by je nebylo vůbec možné řešit. Jako řešení je možné použít výpočetní techniku k reprodukci reálných problémů pomocí modelování a simulace.

Abychom mohli reprezentovat komplexní prostředí, potřebujeme jeho model. Většina problémů je příliš složitá, proto je model pouze přiblížením skutečnosti. Volíme pouze důležité vlastnosti problému, ty méně důležité nezahrnujeme. Tomuto procesu se říká abstrakce. Simulované modely můžeme analyzovat a následně upravovat jejich parametry. Díky těmto úpravám můžeme vytvářet nové modely, které lépe odpovídají našim požadavkům. Díky tomu dostaneme lepší přehled o problému.

Tématem této diplomové práce je dopravní simulace. Modely dopravy nám pomáhají porozumět dynamice pohybů a řízení dopravy. Díky stále větší dopravní hustotě se stává dopravní situace čím dál složitější. Staví se nové křižovatky, nadjezdy, křížení cest a podobně. Tato komplikovaná dopravní situace vyžadují hlubší analýzu, díky které může dojít k zlepšení a optimalizaci. Vysokými požadavky na dopravní síť se vytváří dopravní zácpy. Počítačové simulace jsou vhodné například pro modelování dopravních zácp, vytvořených mezi dvěma křižovatkami kvůli červeným světlům na semaforech. Díky simulaci můžeme analyzovat a vyhodnocovat chování dopravy v křižovatkách, na silnicích a dálnicích. Dle toho potom upravovat parametry (intervaly světel semaforu) a zkoušet rozmanitá řešení. To vše nám poskytuje silný nástroj pro simulaci řízení dopravy.

1.1 Struktura práce

V sekci 2 se seznámíme s modelováním dopravy a pojmem dopravní tok. V sekci 3 je probrán pojem agent, různé typy agentů a jejich využití. 4. sekce se týká softwaru MATSim, jeho architektury. 5. sekce se zaobírá rozšiřováním MATSimu a implementovaním rozšíření. V 6. sekci jsou provedeny pokusy v MATSimu, jejich popis a výsledky. Poslední sekce 7 obsahuje závěr diplomové práce.

2 Modelování dopravy

2.1 Dopravní tok

Modelování dopravy se provádí pomocí dopravních toků. Dle [10] jsou dopravní toky jedním z nejdůležitějších pojmů při modelování dopravy.

Tok je proudem vozidel a dopravy odehrávajícím se v silniční síti. Jsou v něm modelovány vztahy mezi jednotlivými vozidly. Silniční síť se skládá z uzlů a úseků. Úseky spojují jednotlivé uzly.

Shvetsov [21] rozlišuje tři typy modelů pro modelování dopravních toků:

Predikční model: Předpovídá zatížení sítě a dopravních uzlů pomocí průměrných vlastností dopravy, například počet vozidel nebo osob v dopravě.

Simulační model: Slouží k vytváření dopravy v různých měřítcích 2.2.

Optimalizační model: Využívá se k optimalizaci pohybu osob, nákladů a následné úpravy sítě dle výsledku.

Simulace jednotlivých částí dopravního toku se provádí pomocí matematických modelů.[5].

Poptávkový model: Simuluje dopravu z pohledu uživatelů a dopravní infrastruktury. Zohledňuje vnější vlivy, znečištění, dopravní nehody a podobně.

Model zatížení sítě: Je obsažen v poptávkovém modelu. Modeluje zatížení prvků sítě (uzly a úseky) během toku.

Výkonnostně úsekový model: Zabývá se vztahem mezi dobou trvání cesty a hustotou dopravy na úseku od jeho počátku až do konce.

Cestovně-výkonnostní model: Simuluje jednotlivé prvky na cestě od začátku až do konce cesty.

Model požadavků: Slouží k jinému účelu než poptávkový model. Nejčastěji pro analýzu požadavků dopravního systému, určení nastavení těchto požadavků a simulaci různých možností požadavků. Požadavky vznikají na základě potřeby vykonat nějakou činnost na nějakém místě. Model požadavků svazuje tok požadavků k dané činnosti. Například než osoba začne cestu, je třeba vykonat několik rozhodnutí. Rozhodnutí jsou ovlivněna různými parametry, například vlastnictví automobilu, trasa, cíl cesty.

Model přiřazení: Modeluje spolupráci mezi poptávkovými a zásobovacími modely. Díky těmto modelům můžeme sledovat počty uživatelů, výkonnost na jednotlivých úsecích v síti. Jedná se o důležitou součást systému, díky které provádíme ohodnocení celého modelu.

Výše zmíněné modely mají fyzická a funkční omezení – modelování se provádí pouze v části oblasti, model je tedy odstřihnut od okolí.

2.2 Měřítka modelů

Při modelování nás může zajímat jen určitý detail, proto je výhodné mít různá měřítka modelů. Zvyšujícím se měřítkem odpadá potřeba detailních dat a snižuje se i následná doba běhu simulace, nicméně za cenu snížené detailnosti. Dle [10] se pro různé situace využívají tři různá měřítka. Jedná se o makroskopické, mikroskopické a mesoskopické měřítko modelu.

2.3 Makroskopické měřítko

Při využití makroskopického měřítka nejsou v modelu vozidla reprezentována jako jednotlivci, ale jako skupiny[10]. Podobně jako modely simulace kapaliny protékající sítí ulic. Mezi hlavní parametry patří:

Rychlost: Značí se v. Jedná se o vzdálenost, kterou urazí vozidlo za jednotku času. Např. $\mathrm{km\,h^{-1}}$, $\mathrm{m\,s^{-1}}$.

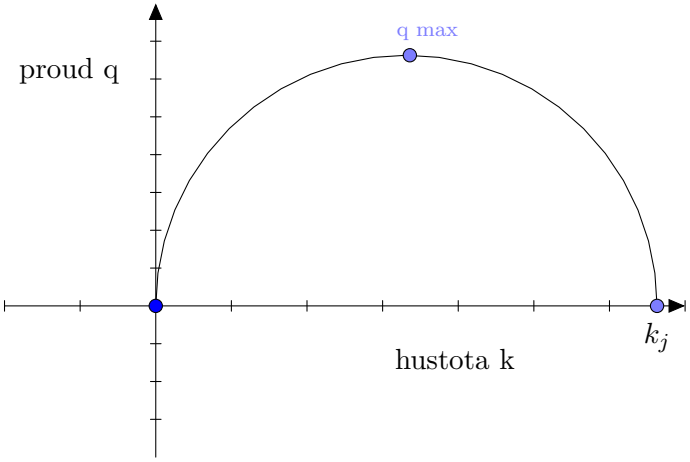
Hustota: Značí se k. Jedná se o počet vozidel na úseku silnice. Např. počet vozidel na jednom kilometru. Mezi důležité hodnoty hustoty patří maximální hustota dopravy k_c, při které je volný průjezd a k_j - hustota, kdy dojde k dopravní zácpě.

Proud: Značí se q. Je to počet vozidel, která projela určitým bodem za jednotku času. Např. počet vozidel za hodinu.

Mezi těmito třemi parametry existuje vztah.

$$q = k * v \tag{1}$$

Parametry lze měřit dvěma způsoby. První možností je stanovení dvou nebo více pevných bodů. Měřený parametr nastane, pokud vozidlo projede měřeným bodem. Druhou možností, časovými body, tedy na jaké pozici se vozidlo nachází v určitou dobu.



Obrázek 1: Vztah mezi hustotou a proudem

Vztah mezi hustotou a proudem je znázorněn na Obrázku 1.

Při nulové hustotě dopravy je i nulový tok. Se zvyšující se hustotou se zvyšuje i tok dopravy. Proud a hustota se zvyšují až po hodnotu q_{max}. Další zvyšování hustoty snižuje proud, až nakonec dojdeme k hodnotě hustoty k_j, tedy proud dopravy se zastaví.

2.4 Mikroskopické měřítko

Zde je dopravní situace modelována na úrovni jednotlivých vozidel a interakce mezi nimi. Jednotlivá vozidla jsou zde jako individuality. Chovají se podle stanovených pravidel. Mikroskopický model poskytuje detailnější informace než makroskopický. Potřebuje pro svojí práci více dat, jejich zpracování je ale výpočetně náročnější. Existuje celá řada mikroskopických modelů. První byl Car-follow model[12], popisující chování řidiče v koloně bez možnosti předjetí. Toto měřítko modelu se hodí pro modelování dopravy v koloně.

2.5 Mesoskopické měřítko

Mesoskopické měřítko modelu kombinuje makroskopické a mikroskopické měřítko. Snaží se využít výhod obou přístupů. Jsou vhodné pro studie nad velkými oblastmi s větší detailností, než kterou nám dopřejí makroskopické modely, ale které by byly příliš náročné pro mikroskopické modely.

3 Agent

Agent je počítačový systém v prostředí, který je schopný autonomního chování, interakce s ostatními agenty nebo jeho prostředím, za účelem vykonání svého cíle[28].

Franklin [9] popisuje prostředí jako dynamický systém. Pro člověka může znamenat planetu Zemi, ale například pro termostat je prostředím místnost, ve které se nachází a vlastností prostředí je teplota. V našem případě se bude jednat v kontextu dopravy o dopravní síť.

Agent přijímá vstupy z prostředí přes senzory, tomuto procesu se říká vnímání (perception), podle svého chování vytváří akce[28]. Agent má pouze částečné informace o svém prostředí a také částečný vliv na prostředí. Má pouze limitovaný pohled. Dvakrát provedená akce nemusí dopadnout stejně i když je provedena za stejných podmínek. Mohou se lišit předchozí zkušenosti agenta a ten se může zachovat pokaždé jinak.

Agent má určitou sadu akcí, kterými ovlivňuje své prostředí. Zmíněné akce nemohou být provedeny v každé situaci, protože některé akce mají určité předpoklady, za kterých mohou být vykonány.

Agenti komunikují s ostatními agenty přes komunikační rozhraní v komunikačním jazyce agentů, jsou schopni spolupracovat a vyměňovat si informace. Dle [19] musí agenti naplňovat dané vlastnosti:

Přizpůsobivost: Agent musí umět pracovat na různých platformách a operačních systémech bez vnějších zásahů.

Mobilita: Agent by se měl dokázat pohybovat v prostředí podle vlastního řízení tak, aby dokázal hledat informace a data pro vykonání svého cíle. Agenti v různých prostředích spolu mohou komunikovat.

Transparentnost a zodpovědnost: Agent se musí chovat vůči svému uživateli transparentně. Podat informace kde se během své cesty nacházel, jaká data sesbíral, s kým se kontaktoval a jaké akce kdy vykonal.

Odolnost: Agent musí být schopen vypořádat se s chybami, neúplnými informacemi a nedostatkem dat i bez zásahu uživatele.

Samostatné uvedení do činnosti: Agenti mají schopnost samostatně se spouštět a zastavovat vždy když shromáždí informace dle uživatelových preferencí v dostatečné frekvenci.

Uživatelsky orientovaný: Agent jedná v zájmu uživatele. Pracuje podle jeho požadavků, ale je schopný přijít i s novým řešením.

V práci [20] přidali další charakteristiku agenta - racionální chování. Racionální agent vybere akci, která je nejcennější podle jeho znalosti prostředí. Kritéria úspěšnosti chování agenta se nazývají výkonnostní měřítka. Agenta můžeme brát jako jednotku v prostředí, ve kterém koná

akce na základě vstupů. Prostředí následně na základě těchto akcí prochází sekvencí stavů. Agent je úspěšný, pokud je tato sekvence stavů žádoucí pro jeho cíl. Výkonnostní měřítka by se tedy měla zaměřit spíše na prostředí, než na chování jednoho agenta. Například nás zajímá, zda v prostředí vzniká dopravní zácpa. Agentova racionalita v dané chvíli záleží na výkonnostních měřítcích, agentově povědomí o okolí, akcích, které může agent provést a agentově dosavadních krocích k dané chvíli. Racionální agent by měl být schopen volit takové akce, aby maximalizoval výkonnostní měřítka.

3.1 Prostředí úkolu

Agenty využíváme k řešení úkolů, které jsou řešeny v prostředí. Prostředí úkolu obsahuje výkonnostní měřítka, prostředí a agenty. Mezi vlastnosti prostředí úkolu patří dle [20]:

Částečné nebo plně zřetelné prostředí: Prostředí je plně zřetelné pokud je agent schopný si obstarat přes své senzory aktuální přesné informace o svém stavu a sledovat všechny stavy prostředí. V částečném prostředí nemůže agent sledovat všechny stavy prostředí.

Pro agenty je jednodušší pracovat v plně zřetelném prostředí, avšak ve skutečnosti bývají prostředí příliš složitá, a proto jsou částečně zřetelná prostředí mnohem častější.

Deterministické nebo stochastické: V deterministickém prostředí je známý další stav. Protože další stav je přesně určen podle stávajícího stavu a předcházejících stavů. Každá akce má svůj přesně daný efekt na prostředí. Ve stochastickém prostředí nelze předpovídat následující stav, je složitější jej modelovat.

Epizodní nebo sekvenční: V epizodním prostředí jsou akce vykonávány atomicky. V epizodě agent vykoná pouze jednu akci. Epizody spolu nejsou propojeny a také nejsou nijak navzájem provázány. Tedy akce vykonané v následující epizodě nesouvisí s akcemi z předchozí epizody. V sekvenčním prostředí všechny předcházející akce ovlivňují všechny budoucí akce. Epizodická prostředí jsou jednodušší než sekvenční, protože agenti nemusí přemýšlet dopředu.

Statické nebo dynamické: Ve statickém prostředí způsobují jeho změny pouze akce agentů. Agent tedy nemusí brát na zřetel prostředí při provádění akce. V dynamickém prostředí probíhají různé procesy, které mohou změnit stavy prostředí během provádění akce agentem. Například semafory.

Diskrétní nebo nepřetržité: Rozdíl mezi diskrétním a nepřetržitým prostředím záleží na stavu prostředí. Jak prostředí pracuje s časem, vstupy a akcemi agentů. Diskrétní prostředí má pevně dány časové intervaly, má konečný počet rozličných stavů a konečný počet vstupů a akcí agentů. U nepřetržitého prostředí se jedná o prostředí, kde jsou stavy a hodnoty nepřetržité, je možné měřit je kvantitativně. Např. jízda na kole reprezentuje nepřetržité prostředí, hra šachů diskrétní.

Jedno nebo multiagentní prostředí: V jedno agentním prostředí operuje pouze jeden agent. U multiagentních prostředí rozlišujeme mezi konkurenčním a spolupracujícím prostředím. V konkurenčním prostředí se agenti snaží maximalizovat vlastní výkonnostní měřítka a minimalizovat výkonnostní měřítka konkurenčních agentů. V spolupracujícím prostředí se agenti snaží maximalizovat společná výkonnostní měřítka.

Rusel a Norvig [20] při výzkumu zjistili, že většina prostředí k modelování je částečně zřetelná, stochastická, sekvenční, dynamická, nepřetržitá a multiagentní. Jedná se tedy o velmi složité a dynamické systémy.

3.2 Softwarový agent

V předchozích sekcích jsme si představili pojem agent a prostředí, jejich vlastnosti. Nyní se budeme zabývat pojmem agent ze softwarového hlediska.

Softwarový agent může být program, komponenta nebo objekt. Na rozdíl od normálních programů jsou agenti aktivní, autonomní, persistentní, reagují na prostředí a jsou orientovaní na cíl.

Dle [4] mají vlastnosti:

Samostatnost: Agenti jsou schopni samostatného chování, úpravy svého plánu za účelem vykonání úkolu od uživatele. Jednotlivé kroky nejsou předem nadefinovány, proto není třeba zásahu uživatele. Agenti by se měli snažit dosáhnout svých cílů i když se mění prostředí a nezkoušet procedury, které již nefungují a nebo ukončit svou činnost pokud již není možné dosáhnout cíle.

Kooperativní chování: Agenti mají schopnost komunikovat s ostatními agenty nebo uživateli. Agenti spolu komunikují přes agentní jazyk pro výměnu informací. V multiagentním prostředí je kooperace klíčová pro dosažení a úspěšné vyřešení zadaného úkolu. Nejedná se pouze o výměnu informací, ale také o vyjednávání s ostatními agenty k dosažení cíle, případně snížení cíle pro dosažení alespoň částečného řešení.

Reaktivita: Agenti vnímají prostředí ve kterém operují přes senzory a korektně reagují na prostředí.

Proaktivita: Agenti nereagují pouze na impulzy z prostředí, ale samostatně provádějí akce a připravují si plán k dosažení cíle.

Persistence: Akce se nevykonávají dle požadavku, ale jsou prováděny v nepřetržitém sledu. Agenti volí, kdy akci provedou.

Proaktivita s reaktivitou jsou důležitější vlastnosti, než se může zdát. Proaktivita je chování vedoucí přímo k cíli. Každá aktivita má nějaké předpoklady a následné důsledky. Aktivita se vykoná, pokud jsou její předpoklady a následky v souladu s cílem. Toto chování, ale není možné

zajistit ve všech prostředích. Pokud se prostředí během vykonávání mění, není možné zajistit, že předpoklady jsou stále stejné, pravděpodobně se změnily. A tedy výsledek akce je nejistý. V multiagentních systémech se nemohou agenti plně seznámit s prostředím, protože prostředí mění několik agentů a vzniká v prostředí nejistota, na kterou musí agenti efektivně reagovat pro dosažení svého cíle. Jak je vidět, není možné použít pouze proaktivní nebo reaktivní agenty. Je třeba využít kombinace obou chování, zajistit správné vybalancování proaktivity s reaktivitou není jednoduché.

3.3 Multiagentní systémy

Multiagentní systém je prostředí, ve kterém pracuje několik vzájemně se ovlivňujících agentů. V našem případě se budeme zabývat softwarovými agenty. Nicméně za multiagentní systémy můžeme považovat například i lidi, případně týmy lidí. Jedná se o simulaci týmové práce. Softwaroví agenti se využívají v případě složitých problémů, které by byly neřešitelné jedním agentem. Problémy jsou tak složité, že přesahují znalosti každého z agentů, a proto je třeba je řešit kooperativně.

Vlastnosti multiagentních systémů[15]:

- Není globální řízení systému.
- Data jsou decentralizovaná.
- Asynchronní provádění výpočtů.
- Agenti pracují autonomně.
- Agenti mají nekompletní informace o prostředí. Samostatně nemají schopnost řešení problému, mají pouze částečný pohled na problém.

Dané vlastnosti zvyšují flexibilitu a přizpůsobitelnost na měnící se okolní prostředí, zvyšují výkonnost.

3.3.1 Umělá inteligence

Multiagentní systémy projevují určitou umělou inteligenci. Dle [28] jsou vhodné pro multiagentní systémy dva druhy umělé inteligence:

Distribuovaná umělá inteligence: Problém se rozdělí na několik dílčích podúkolů, které řeší agenti podle úrovně své kompetence za vzájemné spolupráce s ostatními agenty s cílem dosáhnout řešení problému. Agenti podřizují svou činnost vyřešení problému. Toto se nazývá předpoklad benevolence. Koordinace agentů za účelem splnění podmínky se nazývá kolaborace. Tento přístup se nazývá distribuované řešení problému a spadá do oblasti distribuované umělé inteligence.

Decentralizovaná umělá inteligence: Jak již název napovídá, decentralizovaná umělá inteligence je opakem distribuované umělé inteligence. Agent v multiagentním prostředí již není tak pevně spojen s ostatními agenty, je zaměřen především na řešení svého individuálního cíle. Omezené zdroje a znalosti nutí agenty vytvářet koalice a spolupracující skupiny. Problémem těchto sítí je koordinace přes chybějící centrální řídící mechanismus takto utvořených společenství.

Důležitou vlastností je flexibilita, tak aby mohli být multiagentní systémy použity, upraveny bez zásadních úprav nebo přepisování pro širokou škálu problémů.

3.3.2 Využití multiagentních systémů

Mutliagentní systémy mají širokou škálu využití. Hlavním využitím je řešení problémů, které jsou těžko řešitelné konvenčními způsoby, pomocí jednoho agenta z důvodu omezených zdrojů nebo závislostí na výpočtu jediného systému. Další využití multiagentního přístupu je v případě výhodného propojení nebo spolupráce existujících konvenčních systémů. Při častých změnách je možné zařídit aktuálnost systémů zapouzdřením systému do agenta, který provádí komunikace s ostatnímy agenty. Je možné i využití pro informační zdroje, které jsou prostorově distribuované. Např. různé senzorové sítě, monitorování seismické činnosti nebo získávání informací z internetu.

Využitím multiagentních systémů se zlepšuje výkon, efektivita, spolehlivost, rozšiřitelnost, robustnost, udržitelnost, schopnost reagovat na podněty, flexibilita a znovupoužitelnost.

Multiagentní systémy mají i určité nároky, které znesnadňují jejich návrh. Není snadné přesně formulovat a rozdělit problémy mezi skupiny agentů. Dále je třeba vyhnout se zcela nepředvídatelnému a chaotickému chování agentů díky chybějícímu globálnímu pohledu, který není v multiagentním principu obsažen. Dále mohou být problémové návrhy komunikačních protokolů mezi agenty a zprostředkovávání informací mezi agenty.

Optimálním využitím multiagentních systémů je paralelizace agentů, tím dochází ke zkrácení doby řešení úkolu. Moderní platformy umožňují současný běh statisíců až milionů agentů. Díky tomuto vysokému počtu agentů můžeme porozumět i poměrně složitým a komplexním situacím.

3.4 Mobilní agent

Mobilní agenti přidávají další důležitou vlastnost – mobilitu.

Globální počítačová síť je platforma, ve které jsou vytvářeny a vykonávány procesy agenta, probíhají zde přesuny agenta mezi sítěmi. Místem výkonu agenta je počítač s IP adresou, na kterém se v momentu vykonávání agent nachází.

Mobilní agenti jsou softwarový program, který je schopen se přesunovat v globální počítačové síti[4]. Při přenosu se přenáší stav agenta. Stav agenta reprezentuje jeho stav před tím, než byl agent přenesen do jiné sítě, na jiný počítač. Dle stavu se následně odvíjí další instrukce, které agent vykoná na novém místě. Agent zná svého uživatele a pracuje v jeho prospěch. Proces

přesunu se vykonává bez přičinění uživatele. Mobilního agenta lze jednoznačně identifikovat v rámci globálního prostředí.

Existují dva typy mobilních agentů[16].

Agenti s předdefinovanou cestou: Jejich cesta v síti je předem daná.

Agenti s volnou cestou: Jejich cesta sítí není předem daná. Cesta záleží na aktuálním stavu sítě.

3.4.1 Výhody mobilních agentů

Několik výhod mobilních agentů:

- Mobilní agenti přetváří klient-server volání na přenosná data, předávání mezi-výpočtů je tímto potlačeno. Šetří se kapacita sítě.
- Agenti nepožadují nepřetržité připojení. Uživatel může agenta vypustit do sítě a ukončit své připojení. Opětovné připojení stačí navázat, až když se agent vrátí s výsledkem. Výhodou je vyšší spolehlivost agenta a možnost práce v nestabilní síti.
- Agent pracuje samostatně a asynchronně. Uživatel nemusí monitorovat práci agenta při práci v síti. To šetří čas uživatele, snižuje datové nároky a navíc decentralizuje strukturu sítě.
- Díky inteligenci agenta mohou být informace filtrovány.
- Údržba – pro úpravu úkolu se mění pouze agent, nemusí se měnit jeho prostředí.
- V prostředí může pracovat paralelně několik agentů.
- Větší softwarová přenositelnost, možnost běhu na různých platformách.

4 MATSim

MATSim (Multi-Agent Transport Simulation) projekt byl započat okolo roku 2006 s cílem generování a simulace dopravy, dopravních zácp a sledování jednotlivých cestovatelů (agentů) během jejich denního nebo týdenního plánu činností. Skládá se z kolekce samostatných modulů integrovaných do společného open source frameworku.

MATSim Framework pro implementaci velkých multiagentních dopravních simulací se skládá z několika modulů, které mohou být využity jak samostatně, tak i kombinovaně. Jednotlivé moduly lze nahradit vlastními. Matsim nabízí následující moduly:

- Modelování požadavků agentů.
- Simulace mobility agentů (simulace dopravního toku).
- Přeplánování.
- Ovladač (controler) pro iterativní běh simulací.
- Metody pro analýzu výsledků generovaných moduly.

Framework je navržen pro mikroskopickou simulaci rozsáhlých dopravních scénářů, proto jsou všechny moduly optimalizovány pro efektivní zvládání zamýšlené funkcionality i s pomocí paralelizace.

MATSim je postaven na modelování podle aktivit a to je založeno na poptávkovém modelu2.1. Ústředním prvkem jsou aktivity. Jedná se o činnosti prováděné agentem. Aktivitou může být to, že je doma, v práci, nakupuje a podobně. Aktivity vytváří poptávku.

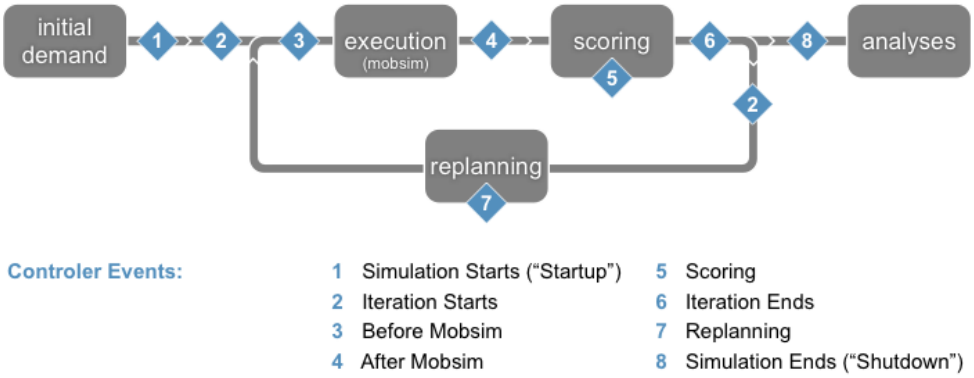
4.1 MATSim simulace

MATSim je navržen pro simulaci jednoho dne, obvyklou jednotkou modelování podle aktivit. Je však možná i vícedenní simulace. Běh MATSimu (provádění simulace) obsahuje nastavitelný počet iterací. Tedy běh MATSimu probíhá v tzv. smyčce na Obrázku 2.

MATSim používá koevoluční princip4.4. Každý agent opakovaně optimalizuje svoje denní aktivity, zatímco soutěží o místo s ostatními agenty v dopravní infrastruktuře.

Nejprve je vytvořena počáteční poptávka (initial demand), vzešlá ze studované populace agentů a jejích denních aktivit v dané oblasti. Chování agentů je odvozeno z empirických dat přes odebírání vzorků modelování chování.

Na počátku je během první iterace poptávka optimalizována pro každého jednotlivého agenta. Každý agent má svou paměť, obsahující pevný počet denních plánů. Denní plán se skládá z řetězce aktivit, cest mezi nimi (tomuto budeme říkat *leg*) a přiděleného ohodnocení. Ohodnocení je možné interpretovat jako ekonometrickou užitečnost. Například užitkové modely[14].



Obrázek 2: Matsim smyčka [18]

V každé iteraci si při načítání dopravní sítě MATSim mobsimem 4.7 (simulace mobility, přesný význam záleží na kontextu. Může se jednat o načítání dopravní sítě, simulaci toku dopravy), každý agent vybere plán ze své paměti. Výběr plánu se řídí podle jeho skóre, které je vypočítáno při každém mobsim běhu (ohodnocení). Základem pro výpočet je výsledek předchozího běhu. Určité části agentů (obvykle 10%) je dovoleno klonovat vybraný plán a upravit tento klon (přeplánování). Při kroku načítání je k dispozici mnoho mobsimů, které lze dále překonfigurovat.

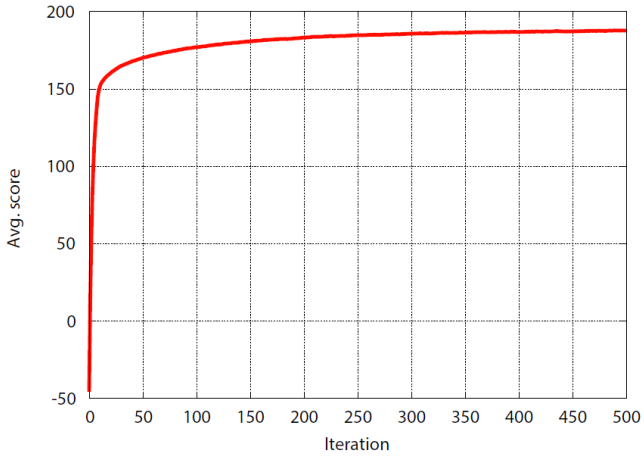
Matsim je postaven na událostech. Události vytváří jednotlivé moduly a všechny akce v simulaci. Každá událost je zaznamenána pro další analýzu. Dané záznamy událostí mohou být seskupeny pro další vyhodnocení v požadovaném měřítku (analýza).

Modifikace plánů (přeplánování) je prováděna pomocí přeplánovacího modulu. MATSim během úpravy zvažuje čtyři dimenze:

- Čas odchodu a trvání aktivity.
- Cestu.
- Typ dopravy.
- Cíl cesty.

Strategie přeplánování a úpravy plánů může být různá. Mutace řešení, přibližné řešení nebo hledání nejlepší možné cesty v každé iteraci. Například nová cesta je většinou určena jako nejlepší možná cesta a její následná úprava, čas odchodu a typ dopravy jsou náhodné mutace.

Pokud se vytvoří příliš plánů pro další postup agenta, pak je plán s nejnižším skóre odstraněn z paměti agenta. V MATSimu je možné výběr plánu dále upravovat vlastními funkcemi. Iterace je ukončena vyhodnocením vybraného denního plánu ohodnocovací funkcí. Iterativní proces je ukončen, jakmile se stabilizuje průměrné skóre populace. Typická křivka vývoje je vidět na Obrázku 3.



Obrázek 3: Vývoj skóre populace[13]

4.2 Controler

Modul controleru (ovladače) zpracovává události vznikající během simulace MATSimu (tzv. MATSim smyčky). Na Obrázku 2 jsou znázorněny různé události vznikající během simulace a řídí samotnou smyčku.

Controler je modul pro řízení běhu MATSimu. Jeho parametry se nastavují v konfiguračním souboru4.5.1. Jedná se například o počet iterací, četnost logování.

4.3 Model toku dopravy v MATSimu

MATSim poskytuje dva interní mobsimy QSim a JDEQSim. Případně je možné připojit externí simulace mobility. Hlavním mobsimem je QSim, převážně díky podpoře paralelního běhu ve více vláknech. V[6] Charypar rozlišuje mezi těmito simulacemi toku dopravy:

Fyzická simulace: Rozlišuje různé detailní modely pohybu aut v dopravě.

Celulární automaty: Zde jsou cesty diskretizovány do buněk.

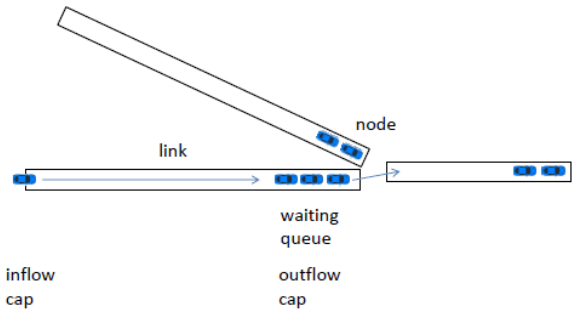
Simulace na základě fronty: Doprava je dynamicky modelována čekajícími frontami.

Mesoskopické modely: Rychlost se určuje pomocí shlukování.

Makroskopický model: V těchto modelech přistupujeme k dopravě jako k tekutině a jejímu toku v síti, nazajímáme se o jednotlivce.

MATSim je navržen pro rozsáhlá území, proto byla zvolena výpočetně efektivní simulace na základě fronty zobrazený na Obrázku4. Auto vstupující na článek sítě (úsek silnice) je přidáno na konec fronty tohoto článku. Auto ve frontě zůstává do té doby, než vyprší čas potřebný pro

volný přejezd úseku, dostane se na začátek fronty a další úsek dovoluje vstup do jeho fronty. Tento přístup je velmi efektivní, avšak má snížené měřítko simulace, není možné zachytit efekt následování vozidla na článku.

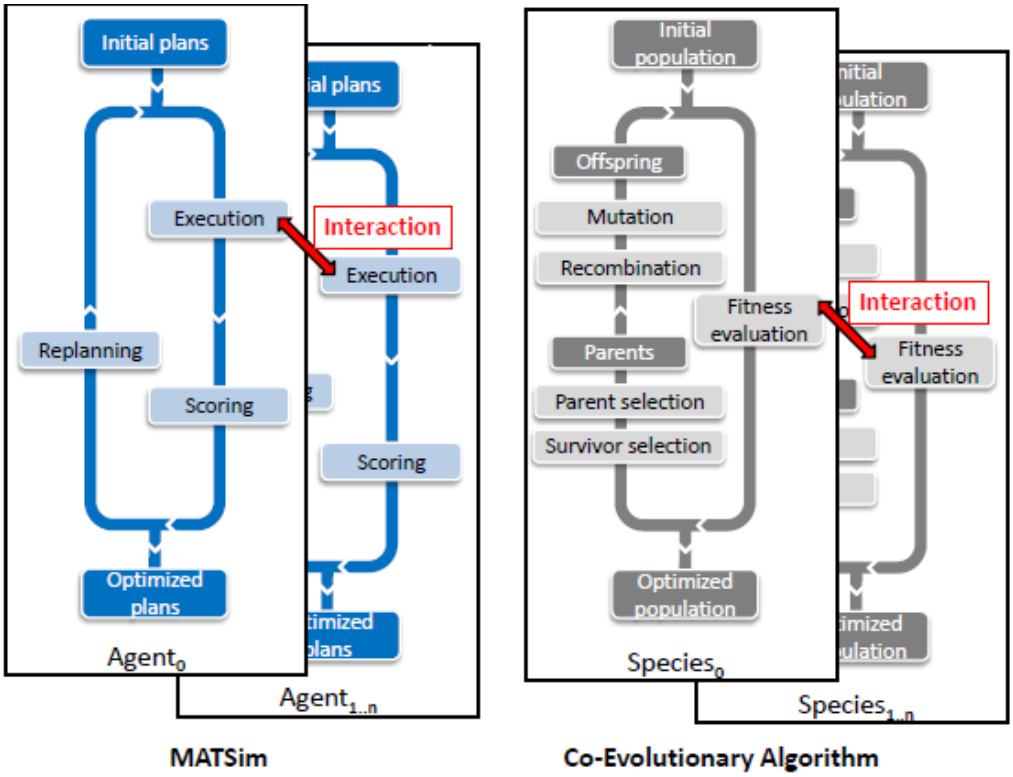


Obrázek 4: Proud dopravy[13]

JDEQSim používá s frontou i události. Agenti neaktualizují svůj postup v pevně daných časových úsecích, ale jen tehdy, pokud mají vykonat nějakou akci. Tedy článek se nemusí zpracovávat během jízdy auta po něm. Spouštění událostí je řízené globálním plánovačem.

MATSim model dopravy je postaven na dvou parametrech. Kapacitě článku – kolik aut se vejde na článek a na kapacitě toku. Kapacita toku specifikuje jeho výstupní kapacitu, tedy kolik cestovatelů může článek opustit během jednoho časového úseku. Vstupní kapacita není v QSimu definována. JDEQSim vstupní kapacitu definuje, díky tomu se mohou zácpy přesunout ze spojení konce prvního článku, kde je generuje QSIM dále na spojení článků, kde by správně měly být. Nicméně pro toto jsou potřebná dodatečná data, která nejsou často k dispozici.

Model toku dopravy byl dále rozšířen řadou dalších modulů: Semafory, modelování více pruhů, interakce mezi různými druhy dopravy.



Obrázek 5: Koevoluční algoritmus[13]

4.4 MATSim Koevoluční algoritmus

Matsim je postaven na koevolučním algoritmu. V těchto algoritmech dochází k vývoji dvou nebo více druhů, při kterém se navzájem ovlivňují. V MATSimu agent reprezentuje druh a chceme aby se vyvíjel jeho plán. Koevoluční algoritmus optimalizuje celodenní plán agenta, jeho aktivity a cesty. Cílem je dosažení rovnováhy agentova plánu. Jakmile je rovnováha dosažena, není možné žádné další vylepšení agentova plánu.

Rozdíl mezi evolučním a koevolučním algoritmem je, že v evolučním algoritmu nás vede algoritmus k celkovému optimu v systému, optimalizace je prováděna globálně v ohodnocovací funkci. U koevolučního algoritmu se snažíme dosáhnout agentova optima. Optimalizace je prováděna na individuálním skóre agenta v rámci jeho plánu.

4.5 Spuštění MATSimu

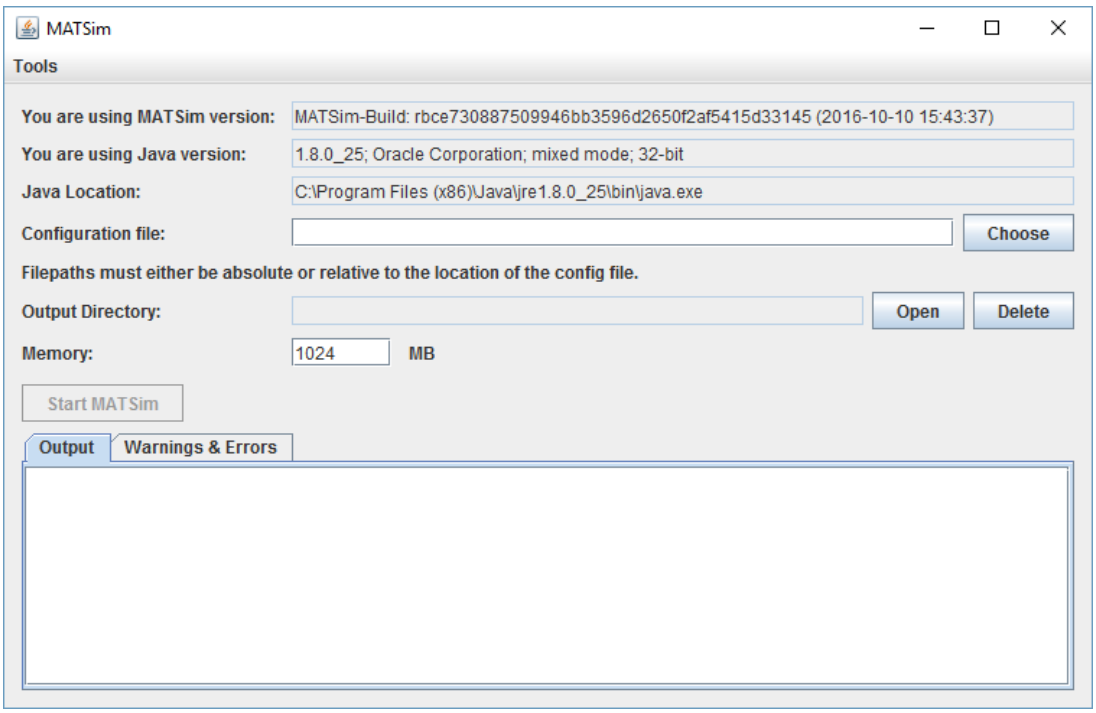
Pro spuštění MATSimu je potřeba nainstalovat Java SE a potřebnou MATSim verzi [1]. V době psaní diplomové práce se jednalo o verzi 0.8.1. Typický MATSim scénář se skládá ze tří souborů:

Konfigurační soubor: Obsahuje konfigurační data MATSimu.

Soubor sítě: Obsahuje popis dopravní sítě.

Soubor populace: Obsahuje informace o agentech, jejich cestovní požadavky a denní plán.

Od verze 0.8.0 obsahuje MATSim uživatelské rozhraní pro spuštění simulace se záložkami pro zobrazení logů a případných chybových hlášení.



Obrázek 6: MATSim GUI

[1] `http://matsim.org/downloads`

4.5.1 Konfigurační soubor

Konfigurační soubor MATSimu je ve formátu xml. Parametry jsou seskupeny do logických skupin. V jedné skupině jsou parametry pro controler jako je počet iterací. Další skupina parametrů je pro mobsim. Uvádí například konečný čas. Konfigurační soubor musí minimálně obsahovat umístění souboru sítě a souboru populace, počet iterací běhu a délku simulovaného dne.

```
<?xml version="1.0" encoding="utf-8"?>
<module name= "global">
   <module name="network">
      <param name= "inputNetworkFile" value = "network.xml"/>
   </module>
   <module name="plans">
      <param name= "inputPlansFile "value="plans.xml"/>
   </module>
   <module name= "controler">
      <param name= "firstIteration" value = "0"/>
      <param name= "lastIteration" value ="30" />
      <param name="outputDirectory" value="./output/kombinace" />
   </module>
   <module name= "planCalcScore">
   <parameterset type= "activityParams">
      <param name= "activityType_0" value = "h" /><!-- home -->
      <param name= "activityTypicalDuration_0" value = "12:00:00"/>
   </parameterset>
</module>
```

Výpis 1: Minimální konfigurační xml soubor

4.6 Sestavení a spuštění scénáře

V této sekci je přehled jednotek, popis potřebných souborů a jejich minimální obsah společně s výslednými soubory MATSim simulace.

Vzdálenost: Vzdálenost se používá pro délku článků. Délka by měla být ve stejných jednotkách, jakou používá souřadnicový systém, aby mohl MATSim vypočítat nejkratší cestu. Nejčastěji používaným souřadnicovým systémem je UTM (Univerzální Mercatorův systém souřadnic[26]), který využívá jako jednotku vzdálenosti metr. Proto je metr nejčastěji využívanou jednotkou vzdálenosti v MATSimu. MATSim očekává použití stejných jednotek v celém scénaři.

Čas: MATSim používá h:min:sec notaci na několika místech, ale vnitřně používá sekundy jako standardní jednotku času. Pro příklad rychlost na článku se udává v jednotce $\mathrm{m\,s^{-1}}$. Výjímkou jsou parametry pro ohodnocovací funkci MATSimu, která používá hodiny.

Peníze: Peníze nemají žádnou předem danou jednotku. Jednotka se nastavuje interně dle země simulace, nebo podle naší volby.

4.6.1 Souřadnicový systém

MATSim nedoporučuje využívat sférický souřadnicový systém [27] (souřadnice od 180° západní délky do 180° východní délky a 90° severní šířky do 90° jižní šířky), protože vypočet vzdálenosti je ve sférickém systému výpočetně velmi náročný.

Místo toho MATSim vypočítává vzdálenost pomocí jednoduché Pythagorovy věty, pro kterou jsou potřeba souřadnice v kartézském systému. Nejlépe tak, aby jedna jednotka odpovídala jednomu metru.

Je doporučeno využívat mapy v UTM[26] souřadnicovém systému. Zde není mapa zobrazena jako celek, ale skládá se ze šedesáti zón zobrazených pomocí transverzálního Mercatorova zobrazení. Jedná se o zobrazení elipsoidu do roviny, a proto lze na mapách v UTM měřit vzdálenost dvou bodů uvnitř zóny pomocí Pythagorovy věty.

4.6.2 Soubor sítě

Soubory jsou ve formátu *xml*. Každý element má své číslo id. Uzly jsou popsány souřadnicemi x a y.

```
<network name="my test network">
   <nodes>
      <node id="1" x="0" y="0"/>
      <node id="2" x="5000" y="0"/>
      <node id="3" x="5000" y="-10000"/>
      <node id="4" x="0" y="-10000"/>
      <node id="5" x="-5000" y="0"/>
   </nodes>
   <links capperiod="01:00:00">
      <link id="1" from="1" to="2" length="5000.00" capacity="36000" freespeed=
         "27.78" permlanes="1" />
      <link id="2" from="2" to="3" length="10000.00" capacity="36000" freespeed
         ="27.78" permlanes="1" />
      <link id="3" from="3" to="4" length="5000.00" capacity="36000" freespeed=
         "27.78" permlanes="1" />
      ...
      <link id="9" from="1" to="5" length="5000.00" capacity="36000" freespeed=
         "27.78" permlanes="1" />
   </links>
</network>
```

Výpis 2: Příklad .xml souboru sítě

Článek má více parametrů:

From a to: Odkazují na uzly, které článek spojuje.

Length: Délka článku, jednotka záleží na zvoleném souřadnicovém systému, typicky se jedná o metry.

Flow capacity: Udává počet vozidel, která přejedou přes článek za hodinu.

Permlanes: Počet pruhů na článku ve směru z „from“ uzlu do „to“ uzlu.

Freespeed: Maximální povolená rychlost vozidla na článku, většinou v $\mathrm{m\,s^{-1}}$.

Modes: Seznam druhů dopravy, kterými je možné cestovat přes článek. Například automobilová nebo veřejná doprava.

Všechny články jsou jednosměrné, pokud je cesta obousměrná, musí být články definovány s opačným pořadím „from“ a „to“ uzlů.

4.6.3 Soubor populace

Soubor populace obsahuje všechny agenty, pohybující se v našem prostředí. Požadavky agenta na cestování jsou popsány v jeho denním plánu.

Soubor *population.xml* populace obsahuje seznam osob a soubor *plans.xml* jejich plány. Soubor populace obsahuje hierarchická data osob. Každá osoba má přiřazen unikátní identifikátor id a seznam plánů.

Agentův plán je popsán seznamem aktivit a *leg*. Aktivita má přiřazen typ a konečný čas, kdy se má v denním plánu vykonat. Poslední aktivita nemá přiřazen čas, automaticky končí počátkem první aktivity v denním plánu. Některé aktivity mohou mít přiřazenou dobu trvání místo konečného času. Jedná se o aktivity automaticky generované plánovacím algoritmem. To, kde bude aktivita vykonávána je dáno přiřazenými souřadnicemi x a y, nebo přiřazeným článkem, který popisuje z jakého článku může být aktivita dosažena. Simulace pracuje s články. Pokud není článek udán, je v případě potřeby vypočítán controlerem nejbližší článek.

Leg nám popisuje, jak agenti cestují z jednoho místa na druhé. *Leg* má přiřazen druh dopravy. Volitelně může mít parametr trav_time popisující dobu přesunu. Pro simulace *leg* je potřeba mít přiřazenou cestu. Její formát závisí na druhu dopravy. Pro auto se jedná o seznam článků, po kterých se musí v daném pořadí přesunout. *Transitleg* obsahuje pouze konečnou lokaci a očekávaný způsob dopravy. Pokud nejsou u počátečního plánu vypočítány cesty, MATSim je dopočte.

Agent vybere jeden plán se seznamu, který je následně vykonán mobsim simulací. Během přeplánovací fáze může být vybrán jiný plán. Plán obsahuje skóre, které bylo vypočteno během simulace mobsimem ohodnocovací funkcí.

```
<plans>
   <person id="1">
      <plan>
         <act type="h" link="9" end_time="05:59" />
               <leg mode="car">
            </leg>
            <act type="w" x="0" y="-10000" dur="02:30" />
               <leg mode="car">
            </leg>
         <act type="h" link="9" />
      </plan>
   </person>
   <person id="2">
      <plan>
         <act type="h" link="9" end_time="07:30" />
            <leg mode="car">
            </leg>
   ...
</plan>
</person>
</plans>
```

Výpis 3: Příklad .xml souboru populace

Shrnutí pro populační soubor:

- Každá osoba má minimálně jeden plán.
- Plán nemusí být vybrán nebo mít skóre.
- K umístění aktivity stačí pouze souřadnice, není třeba uvádět přesný článek.
- Aktivity by měly mít rozumný konečný čas.
- *Leg* potřebuje alespoň způsob dopravy, nemusí mít cestu.

4.6.4 Výstupní soubory

MATSIM vytváří výstupní soubory pro sledování a monitorování simulace, případně vyhodnocení jejího výsledku. Některé soubory, jako například skórovací přehled, jsou vytvářeny implicitně, jiné je třeba povolit pomocí konfiguračního souboru.

Soubory popisující běh MATSimu se ukládají do *output* složky. Soubory vztahující se k iteracím se ukládají do složky $ITERS/it.$*číslo iterace*, které jsou vytvářeny v *output* složce. Četnost

vytváření souborů může být specifikována v konfiguračním souboru. Iterační soubory jsou ukládány po každé iteraci.

Výstupní soubory:

Log soubor: Záznamy potřebné pro naší případnou další analýzu. Záznamy v případě pádu.

Varování a chybové log soubory: Pokud MATSim zjistí problém v simulaci nebo konfiguraci, zapíše chybová hlášené do souboru. Protože log soubor by byl příliš velký, jsou chybová hlášení zapisována do zvláštního souboru pro chybová hlášení.

Skóre statistika: Hodnoty průměrně nejlepšího, nejhoršího a průměrného skóre plánů všech agentů v jednotlivých iteracích. Je vytvářen textový soubor *scorestat.txt*, ale i grafické ztvárnění *scorestat.png*.

Leg Travel Distance statistika: Soubor obsahuje průměrné vzdálenosti z *leg*. Hodnoty jsou opět ukládány jak v textové *traveldistancestats.txt*, tak i grafické podobě *traveldistancestats.png*.

Stopwatch: Strojový čas akcí, jako je přeplánování nebo provádění aktivit v mobsimu v každé iteraci. Data jsou vhodná k analýze výpočtů. Například jak dlouho trvá přeplánování.

Soubory pro iterace:

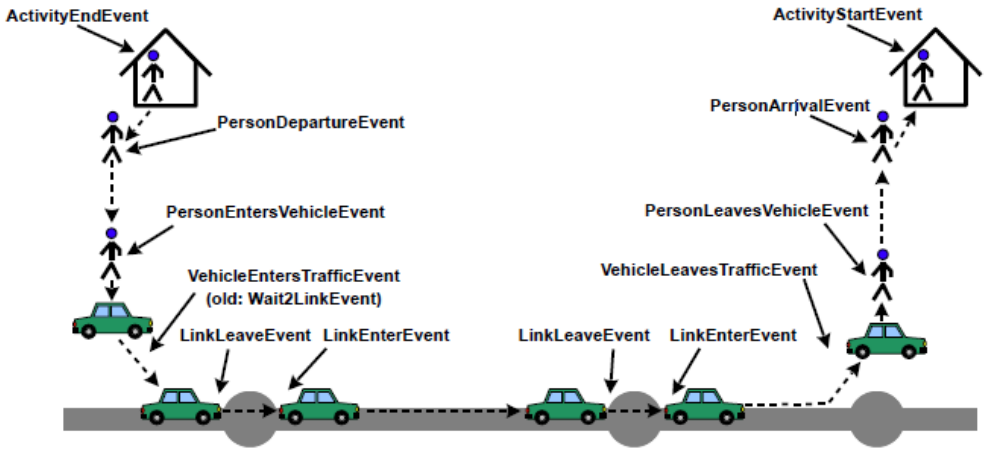
Události: Každá akce v simulaci se ukládá jako událost. Událost má několik parametrů. Id agenta, který způsobil událost, nebo článek na kterém se událost stala. Čas provedení události. Události se využívají v následné analýze po simulaci. 7

Plány: Aktuální stav plánů populace. Plán v konečné iteraci je vygenerován na vrchol výstupní složky.

Leg Histogram: Pro každou iteraci je nakreslen *leg* histogram. *Leg* histogram zobrazuje počet příchozích, odchozích nebo cestujících agentů během časové jednotky. Histogram je vytvořen pro každý způsob dopravy ve tvaru: 1.*legHistogram_car.png* nebo 1.*legHistogram_all.png*. Je vytvořen i souhrnný přehled pro všechny způsoby dopravy. Opět jsou vytvořeny i textové výstupy. 1.*legHistogram.txt*.

Trip duration: Pro každou iteraci ukládá seznam cest a jejich délku trvání do souboru 1.*tripdurations.txt*. Soubory vytváří pro seskupení skládající se z dvou aktivit. Tedy např. pro cestu z domu do práce, z domu na nákupy.

Link stats: Informace o článku, jako je počet vozidel za hodinu na článku, průměrná doba trvání přejezdu článku. Tato data jsou důležítá pro porovnání s reálnými daty.



Obrázek 7: Mobsim události[13]

4.7 Ohodnocovací funkce

MATSim je postaven na koevolučním algoritmu. Každý agent se učí udržováním několika plánů, které jsou hodnoceny a vyhodnocovány v mobsimu, následně vybrány podle skóre nebo upravovány. Ohodnocovací funkce musí být správná pro naše prostředí.

Tento iterativní proces obsahuje následující elementy:

Mobsim: Simulace mobility vybere jeden plán pro agenta a vyhodnotí ho v prostředí. Tomuto se říká načítání sítě.

Ohodnocení: Vybraný plán je ohodnocen pomocí hodnotící funkce.

Přeplánování: Skládá se z několika kroků:

1. Pokud má agent více plánů, než je maximální možný počet (nastaveno v konfiguračním souboru), pak jsou přebývající plány odebrány podle nastavitelného výběru plánu.
2. Někteří agenti mají plán zkopírován, upraven a následně vybrán pro další iteraci. (úprava plánu)
3. Ostatní agenti vyberou plán mezi svými plány (výběr plánu).

Výběr plánu agenta v dané iteraci se skládá z množiny kroků pro výběr. Kroky 1 a 2 modifikují plány při přeplánování a upravují výběr. V 3. kroku je implementován samotný výběr z plánů.

Všechny tří kroky přeplánování by měly směřovat k lepším výsledkům. Odstraněny by měly být jen ty špatné plány. Upravovací krok vytvářet lepší plány a vybírací krok vybrat dobrý plán, který bude mít dobré skóre v mobsim simulaci. Protože se jedná o složitý proces a například

upravovací krok nemusí vždy vytvářet lepší skóre, je vhodné, aby více než jedno řešení bylo dobré a vedlo k vysokému skóre.

Z výše uvedeného je jasné, že hodnotící funkce je důležitým prvkem simulace v MATSimu. Pouze řešení s vysokým ohodnocením budou vybrána a vyhnou se odstranění v kroku odstraňování. Plány s vysokým ohodnocením by si měly vést dobře i v simulaci a naopak plány s nízkým ohodnocením by neměly vyhovovat. Nakolik je plán dobrý nebo špatný záleží na preferencích agenta. Někdo může upřednostňovat dopravu vlastním vozidlem, jiný levnější veřejnou dopravou a někdo jiný cestu na kole.

Abychom překonali tento rozdíl, využívají se v MATSimu funkce ekonometrické užitečnosti. Ty vyjadřují míru spokojenosti přes spotřebu zboží. Například užitkové modely [14] pro ohodnocení. V umělé inteligenci mohou být funkce užitečnosti použity v obecnější rovině pro ohodnocení agenta. V MATSimu můžeme termíny užitečnost a ohodnocení vzájemně zaměňovat. Dále budeme používat termín mezní užitek, protože je nezvyklé používat termín mezní ohodnocení.

Mezní užitek vyjadřuje o kolik vzroste celkový užitek, pokud se množství spotřebovávaného statku zvýší o jednotku. Je odvozen z celkového užitku. Mezní veličiny obecně vyjadřují přírůstky a udávají o kolik se změní jedna proměnná při změně druhé proměnné o jednotku[25].

4.7.1 Charypar-Nagel užitková funkce

První základní ohodnocovací funkce v MATSimu byla formulována Charyparem a Nagelem [11]. Funkce je založena na Vickreyho modelu přeplněných silnic z roku 1969 [22].

Základní funkce počítá užitečnost plánu S_{plan} jako sumu všech užitečností aktivit $S_{act,q}$ plus sumu všech užitečností aktivit cestovatele $S_{trav,mode(q)}$.

$$S_{plan} = \sum_{q=0}^{N-1} S_{act,q} + \sum_{q=0}^{N-1} S_{trav,mode(q)} \tag{2}$$

kde N je počet aktivit. q je cesta, která následuje aktivitu q. Při hodnocení je poslední aktivita spojena s první, aby byl stejný počet cest a aktivit.

Užitečnost aktivity je vypočítána jako:

$$S_{act,q} = S_{dur,q} + S_{wait,q} + S_{late.ar,q} + S_{early.dp,q} + S_{short.dur,q} \tag{3}$$

Jednotlivé členy znamenají:

- $$S_{dur,q} = \beta_{dur,q} * t_{typ,q} * ln(t_{dur,q/t_0,q}) \tag{4}$$

  Jedná se o užitečnost provádění aktivity q, pokud musíme brát v potaz otevírací dobu aktivity v jejím místě. $t_{dur,q}$ je délka vykonávání aktivity. $\beta_{dur,q}$ se vztahuje k meznímu užitku během vykonávání aktivity (nebo meznímu užitku času jako zdroje, toto je stejné pro všechny aktivity). t_0, q je doba, kdy začne být užitečnost pozitivní.

- $$S_{wait,q} = \beta_{wait,q} * t_{wait,q} \tag{5}$$

  Představuje dobu čekání. Například doba čekání před stále zavřeným obchodem. $\beta_{wait,q}$ je tzv. přímý mezní užitek času stráveného čekáním. $t_{wait,q}$ je doba čekání. Je doporučeno nechat $\beta_{wait,q}$ na hodnotě 0.

- $$late.ar, q = \begin{cases} \beta_{late.ar} * (t_{start,q} - t_{latest,q}) & if t_{start,q} > t_{latest,q} \\ 0 & else \end{cases} \tag{6}$$

  Tato rovnice má význam zachycení pozdního příchodu a jeho penalizace. $t_{start,q}$ je začátek aktivity q a $t_{latest,q}$ je posdní čas kdy může aktivita začít bez postihu (například začátek filmu v kině).

- $$S_{early.dp,q} = \begin{cases} \beta_{early.dp} * (t_{end,q} - t_{earliest.dp,q}) & if t_{end,q} > t_{earliest.dp,q} \\ 0 & else \end{cases} \tag{7}$$

  Tato rovnice má význam penalizace za příliš brzké opuštění. $t_{end,q}$ vyjadřuje konec aktivity a $t_{earliest.dp,q}$ vyjadřuje nejdříve možný konečný čas aktivity q. $\beta_{early.dp}$ se použije pouze tehdy, pokud jsou známa dobrá data, jinak je doporučeno ji nechat jako 0.

- $$S_{short.dur,q} = \begin{cases} \beta_{short.dur} * (t_{short.dur,q} - t_{dur,q}) & if t_{dur,q} < t_{short.dur,q} \\ 0 & else \end{cases} \tag{8}$$

  Tato rovnice má význam penalizace pro příliš krátkou aktivitu. $t_{short.dur,q}$ je nejkratší možná doba trvání aktivity. $\beta_{short.dur}$ se použije pouze tehdy, pokud jsou známa dobrá data, jinak je doporučeno ji nechat jako 0.

Výše uvedené hodnoty jsou zadány v configu takto:

```
...
   <module name="planCalcScore">
      <param name="learningRate" value="1.0" />
      <param name="BrainExpBeta" value="2.0" />
      <param name="lateArrival" value="-18" />
      <param name="earlyDeparture" value="-0" />
      <param name="performing" value="+6" />
      <param name="traveling" value="-6" />
      <param name="waiting" value="-0" />

      <param name="activityType_0"        value="h" /> <!-- home -->
      <param name="activityPriority_0"    value="1" />
      <param name="activityTypicalDuration_0" value="12:00:00" />
      <param name="activityMinimalDuration_0" value="08:00:00" />
   ...
```

Výpis 4: Příklad .xml souboru nastavení ohodnocovací funkce v konfiguračním souboru

Škodlivý účinek cestování pro *leg* q je dán:

$$S_{trav,q} = C_{mode(q)} + \beta_{trav,mode(q)} * t_{trav,q} + \beta_m * \Delta m_q + (\beta_{d,mode(q)} + \beta_m * \gamma_{d,mode(q)}) * d_{trav,q} + \beta_{transfer} * \chi_{transfer,q} \tag{9}$$

kde jednotlivé prvky znamenají:

$C_{mode(q)}$**:** Konstanta specifického druhu dopravy.

$\beta_{trav,mode(q)}$**:** Přímý mezní užitek cestování daný druhem dopravy. MATSim používá pro hodnocení 24 hodinové epizody, jedná se o další mezní užitečnost k mezní užitečnosti času jako zdroje.

$t_{trav,q}$**:** Doba cesty mezi místem aktivity q a q+1.

β_m**:** Mezní užitek peněz.

Δm_q**:** Změna v rozpočtu způsobená platbou poplatků, jako je mýtné, průjezd zpoplatněným úsekem (je 0 nebo negativní hodnota, protože utrácíme peníze).

$\beta_{d,mode(q)}$**:** Mezní užitečnost vzdálenosti (0 nebo negativní).

$\gamma_{d,mode(q)}$**:** Zpoplatnění dle druhu dopravy (0 nebo negativní).

$d_{trav,q}$**:** Je vzdálenost mezi místem aktivity q a q+1.

$\beta_{transfer}$: Penalizace za využití veřejné dopravy (většinou negativní hodnota).

$\chi_{transfer,q}$: 0 nebo 1, značí zda byl proveden přesun mezi nynější a předchozí *leg*.

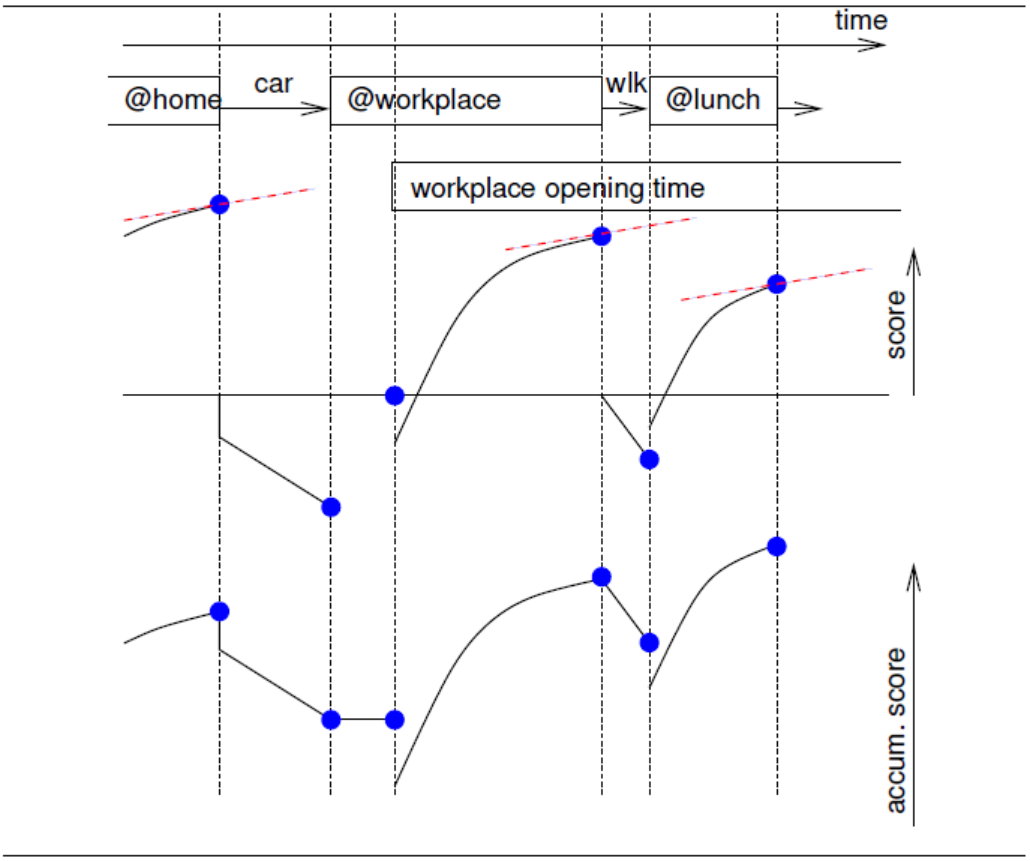
```
<parameterset type="scoringParameters" >
   <param name="earlyDeparture" value="-0.0" />
   <param name="lateArrival" value="-18.0" />
   <param name="marginalUtilityOfMoney" value="1.0" />
   <param name="performing" value="6.0" />
   <param name="subpopulation" value="null" />
   <param name="utilityOfLineSwitch" value="-1.0" />
   <param name="waiting" value="-0.0" />
   <param name="waitingPt" value="-6.0" />

   <parameterset type="activityParams" >
      <param name="activityType" value="car interaction" />
```

Výpis 5: Příklad .xml souboru nastavení ohodnocovací funkce

Povšimněme si, že vzdálenost se projevuje v škodlivosti cestování dvěma způsoby. První je v $\beta_{d,mode(q)}$, tedy tam, kde je potřeba vyvinout fyzické úsilí - chození, jízda na kole. Druhý je $\beta_m * \gamma_{d,mode(q)}$, což je pro automobilovou dopravu, kde větší vzdálenost také zvyšuje výdaje na cestu (větší spotřeba paliva).

4.7.2 Příklad

Na obrázku 8 je příklad hodnotící funkce. Čas je znázorněn směrem zleva doprava. Příklad zobrazuje část vykonaného rozvrhu s aktivitami - *domov*, *práce* a *oběd* spojených cestou autem a chůzí. Aktivity jsou ohodnoceny konkávními funkcemi, čím déle aktivitu provádíme, tím více se zvyšuje ohodnocení. Ohodnocení cestování je naopak modelováno rovnou klesající přímkou. Její sklon je určen druhem dopravy.



Obrázek 8: Zobrazení charakteru hodnoticího procesu. Nahoře jsou jednotlive aktivity, přesuny(*leg*) a jejich přespění k ohodnocení. V dolní části je součet ohodnocení[13].

Za povšimnutí stojí interval mezi příjezdem do práce a začátkem pracovní doby, kdy se neakumuluje žádné ohodnocení.

Maximálního ohodnocení aktivit agentů dosáhneme, pokud mají aktivity stejný mezní užitek (červené linky). Toto pochází ze základů ekonometrické teorie. Pokud by neměly všechny stejný sklon, pak by mohl agent navýšit své skóre prudším sklonem na úkor ostatních aktivit. Tohle platí jen tehdy, pokud ostatní konstanty jako je například cestovní čas zůstanou stejné.

4.7.3 Zaobalovací princip

Hodnotící funkce předpokládá, že plány jsou zacykleny do 24 hodin. Zaobalí je dohromady. Tedy poslední aktivita je spojena s první do jedné aktivity. Například pokud první aktivita začíná ráno v 8 hodin a poslední aktivita začíná v 11 večer, pak se předpokládá, že se jedná o jednu a tu samou aktivitu s dobou trvání 9 hodin.

4.8 Časové náklady obětované příležitosti

Při zaobalovacím principu dostane cestování vedle přímého negativního mezního účinku $\beta_{trav,mode}$ ještě další přímou penalizaci mezního účinku času jako zdroje (ztrácíme čas cestováním). Pokud může být čas cestování snížen o Δt_{trav}, osoba bude prosperovat nejen z vyhnutí se β_{trav} * Δt_{trav}, ale i z přidaného času pro aktivity. Říká se tomu časové náklady obětované příležitosti (opportunity cost of time)[24].

Celková mezní účinnost šetření cestovních nákladů $mUTTS$ je:

$$mUTTS = \frac{\partial}{\partial t_{trav}} S_{trav} + \frac{\partial}{\partial t_{dur}} S_{dur} \tag{10}$$

A to je:

$$mUTTS = -\beta_{trav} + \beta_{dur} * \frac{t_{typ,q}}{t_{dur,q}} \tag{11}$$

Typická doba trvání aktivity je:

$$mUTTS = |_{t_{dur,q}=t_{typ,q}} = -\beta_{trav} + \beta_{dur} \tag{12}$$

$mUTTS$ můžeme definovat jako nepřímý efekt na celkový čas, korigován offsetem β_{trav}, který určuje, jak je dobré nebo špatné, cestovat než nic nedělat. Tedy zda je lepši čekat nebo jet spojem s delší trasou.

Mezní užitek ušetřeného cestovního času může být vyjádřen pomocí VTTS (value of travel time savings[8]), tak že ho rozdělíme na mezní užitek peněz β_m.

$$VTTS = \frac{mUTTS}{\beta_m} = \frac{\beta_{trav} + \beta_{dur} * \frac{t_{typ,q}}{t_{dur,q}}}{\beta_m} \tag{13}$$

a typické trvání aktivity:

$$VTTS = |_{t_{dur,q}=t_{typ,q}} = \frac{mUTTS}{\beta_m}|_{t_{dur,q}=t_{typ,q}} = \frac{\beta_{trav} + \beta_{dur}}{\beta_m} \tag{14}$$

Jedná se tedy o hodnotu času trávenou cestováním. Toto je důležite pro kalibraci účinkové funkce.

4.8.1 Modelování neplánovaného příchodu

Brzký příchod - mezní účinek ušetřeného času na cestě je dán nejen jako $-\beta_{trav}$, ale i jako $-\beta_{trav} + \beta_{dur} * \frac{t_{typ,q}}{t_{dur,q}}$.

Mezní užitek ušetřeného času při čekání je dán jako: $mUWTS = -\beta_{wait} + \beta_{dur} * \frac{t_{typ,q}}{t_{dur,q}}$.

I když se přímý užitek čekání β_{wait} rovná nule, tak i nicnedělání stále ubírá z vymezeného času, tedy vzniká stejná ztráta času (časové náklady obětované příležitosti) jako při cestování. Z toho vyplývá, že abychom mohli déle čekat, musíme předcházející aktivitu opustit dřív a tím

snížíme její ohodnocení. Z toho plyne, že pokud nedokážeme odhadnout β_{wait} odděleně od β_{dur}, je doporučeno nechat β_{wait} na nule.

4.8.2 Pozdní příchod

Mezní účinek pozdního příchodu se značí jako β_{late}, jedná se o záporné číslo. Není zde žádný další mezní užitek, protože pokud přijdeme později, znamená to, že jsme předchozí aktivitu opustili později (a naakumulovali z ní větší skóre) a i nynější začínáme o stejnou dobu později, čímž se nemění výsledné skóre.

4.9 Implementace

V této sekci jsou shrnuty některé detaily iplementace MATSim ohodnocovací funkce. Specifické případy jako je nulová užitečnost, záporná doba trvání aktivity, průměrování ohodnocení, vynucené směřování ohodnocení a typické nastavení hodnotících parametrů.

4.9.1 Nulová užitečnost

Pokud má aktivita užitečnost 0, pak je její doba trvání vypočítána jako:

$$t_{0,q} = t_{typ,q} * exp\left(- \frac{10h}{t_{typ,q} * prio} \right). \tag{15}$$

prio je nastavitelný parametr tak, aby všechny aktivity, které ho využijí dostaly stejnou hodnotu. Když se $t_{dur,q} = t_{typ,q}$, pak je hodnota rovna $10 * \beta_{dur}$. Tato myšlenka ji dělá stejně ohodnocenou jako je zrušení aktivity z nedostatku času.

Bohužel tato myšlenka nefunguje tak jak byla zamýšlena, protože aktivity ohodnocené během krátké doby dostanou větší kumulativní užitečnost za jednotku času než ostatní, takže jsou zrušeny později. Je doporučeno:

- Nemodifikovat *prio* hodnotu na jinou než výchozí.
- Rozeznat, že nynější používaná hodnotící funkce není vhodná pro zrušení aktivit.

4.9.2 Záporná doba trvání aktivity

Při zaobalování (spojení první aktivity s poslední aktivitou) může nastat situace, kdy aktivita trvá zápornou dobu. Například agent má zůstat doma do 3 hodin ráno, pak vykoná svůj denní plán, kde je na konci dlouhá oslava, např. do 6 hodin ráno. Při tomto zaobalení ale dojde k době trvání aktivity -3h.

Původně se při tomto jevu přiřadila aktivitě užitečnost 0. Nicméně tohoto využili agenti pro svoji výhodu. Rozšiřování záporného trvání aktivity bez penalizace, vedlo k více času pro ostatní aktivity, kdy agent mohl naakumulovat více ohodnocení. Proto je třeba nastavit pro daný případ penalizaci.

Jak je vidět, pokud nejsou první a poslední aktivita stejné, může to vést k různým problémům.

4.9.3 Průměrování ohodnocení

Skóre se nepřiřazuje přímo, ale ještě před tím je exponenciálně vyhlazeno:

$$S^k = \alpha S + (1 - \alpha)S^{k-1} \tag{16}$$

S^k je aktuální ohodnocení, S^{k-1} je předchozí ohodnocení, S je ohodnocení ze simulace mobsimu, α je učební koeficient a nastavuje se v konfiguračním souboru. Nevyhodnocené plány si dále udržují svoje ohodnocení.

4.9.4 Vynucení směřování ohodnocení

Pro mnohé situace je vhodné, aby ohodnocení směřovalo k očekávané hodnotě. Průměrovací rovnice nám toto nezajistí, pouze omezuje velké rozpětí ohodnocení. MATSim využívá MSA metodu [17], ta je dána vzorcem:

$$S^m = \frac{1}{m}S + \frac{m-1}{m}S^{m-1} \tag{17}$$

Zde je učební koeficient α nahrazen proměnnou $\frac{1}{m}$. m není iterační číslo, ale kolikrát byl plán vykonán a ohodnocen, protože v MATSimu nejsou plány vykonány a hodnoceny v každé iteraci. Tento parametr je možné nastavit v konfiguračním souboru.

4.9.5 Typické nastavení hodnotících parametrů

Defaultní parametry v MATSimu:

β_m	=	1	užitek/peněžní jednotka
β_{dur}	=	6	užitek/h
$\beta_{trav,mode}(q)$	=	-6	užitek/h
β_{wait}	=	0	užitek/h
$\beta_{short.dur}$	=	0	užitek/h
$\beta_{late.ar}(q)$	=	-18	užitek/h
$\beta_{early.dp}$	=	0	užitek/h

Hodnoty parametrů jsou postaveny na bottleneck modelu [2].

V mnoha systémech, které modelujeme, není cestování o nic výhodnější než nicnedělání. Protože přímý mezní užitek cestování β_{trav}, je blízký nule nebo někdy dokonce mírně negativní, je možné toto zohlednit kalibrací parametrů.

- Nastavení β_m (mezního užitku peněz) na jakoukouliv námi preferovanou hodnotu. Tato hodnota by měla být pozitivní, protože mít více peněz zvyšuje užitečnost.

- β_{dur} mezní užitek provádění aktivity je většinou pozitivní hodnota, protože provádění aktivity zvyšuje užitečnost.
- $\beta_{tt,car}$ mezní užitek cestování nastavit na 0. I když je nastaven na 0, je cestování trestáno spotřebovávaným časem na cestování (časovými náklady obětované příležitosti). Pokud cestujeme autem, nemůžeme vykonávat aktivitu a snižuje se nám doba pro β_{dur}.
- Nastavit všechny mezní užitečnosti podle druhu relativního k hodnotě cestování autem. Např. pokud má náš model $-6/h * tt_{car} - 7/h * tt_{pt}$... pak nastavíme $\beta_{dur} = 6, \beta_{tt,car} = 0, \beta_{tt,pt} = -1$ Pokud nemáme jiný druh dopravy, nastavíme všechny $\beta_{tt,mode}$ na nulu.(tedy stejně jako auto).
- Nastavení cestovních výdajů $monetaryDistanceRate$ na příhodné hodnoty. Hodnoty musí být negativní, protože vzdálenost spotřebovává peníze.
- C_{mode} Nastavení dělby přepravní práce dle dostupných dat.

5 Rozšiřování MATSimu

V rámci implementační části práce jsme se zaměřili na to, abychom prezentovali co to znamená přidát rozšíření do programu MATSim. Před samotným přidáním nové funkcionality do prostředí MATSim jsme se museli s tímto prostředím seznámit a připravit potřebné prostředí jak pro impelemntaci, tak pro testování. Proto se v této sekci zaměříme na popis rozšíření, ale také na konfiguraci a přípravu prostředí pro experimenty.

Budeme se zabývat problematikou přípravy prostředí pro vývoj rozšíření MATSimu, dále instalací MATSimu. Poté si přiblížíme co musíme provést a nakonfigurovat, abychom spustili simulaci MATSim nad reálným příkladem. V poslední sekci se budeme zabývat samotnou implementací nového rozšíření.

5.1 Nástroje

Pro programování pro MATSim a jeho rozšiřování budeme potřebovat několik nástrojů. Předně potřebujeme Java vývojové prostředí. Přímo na stránkách MATSimu je doporučeno vývojové prostředí Eclipse IDE - verze MARS [2]. Java SDK verze 1.7 [3] nebo novější (pro main MATSim distribuci stačí verze 1.7). Dále doporučujeme textový editor jako je PSPad [4] pro úpravu xml souborů.

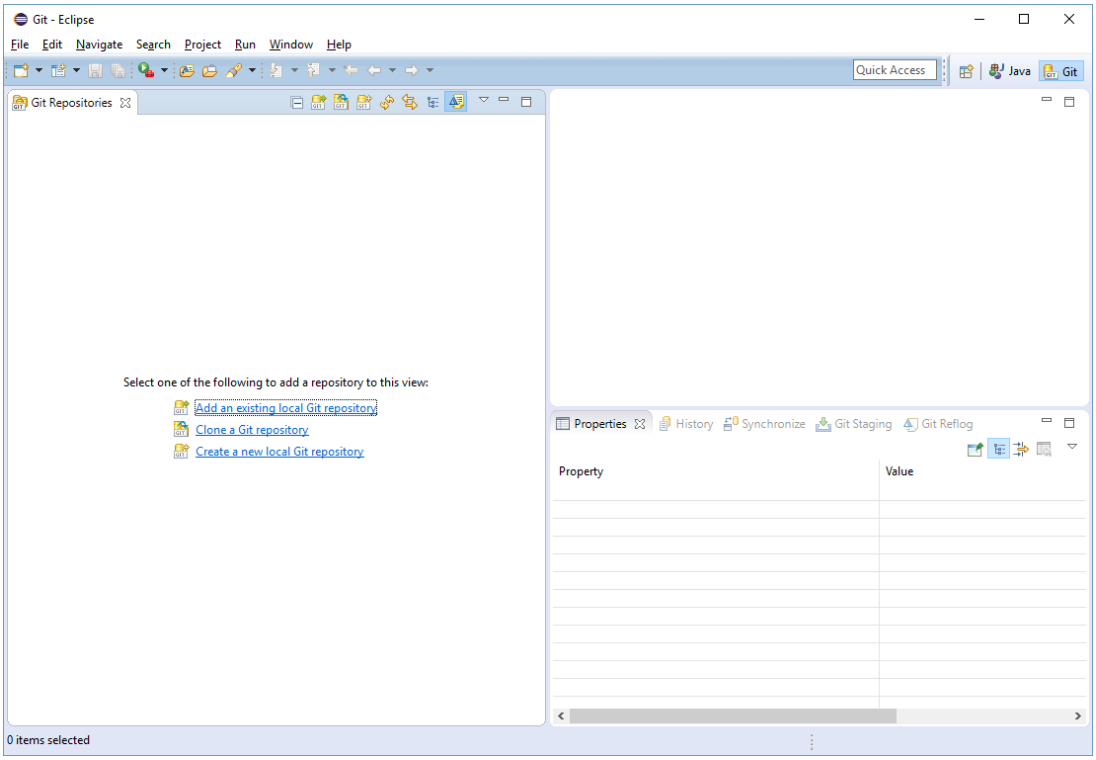
MATSim můžeme přímo stáhnout jako jar soubory z MATSim stránek. V našem případě ale chceme rozšiřovat MATSim funkcionalitu, proto si stáhneme projekt z GitHub uložiště.

5.2 Instalace MATSimu z Git uložiště

Abychom mohli do MATSimu vyvíjet, je potřeba nejprve stáhnout zdrojové soubory. Nejsnadnější cestou je stažení souborů z Git uložiště přímo přes vestavěného klienta ve vývojovém prostředí Eclipse. Nainstalujeme vývojové prostředí Eclipse IDE. Jedná se o klasickou instalaci programu a není tedy důvod jí detailněji rozepisovat. Níže je popsáno stažení z Git uložiště:

- Spustíme Eclipse a vytvoříme nový *workspace*.
- Otevřeme *workbench*.
- Nahoře vpravo přidáme nový *perspective* a v okně zvolíme položku *Git*.

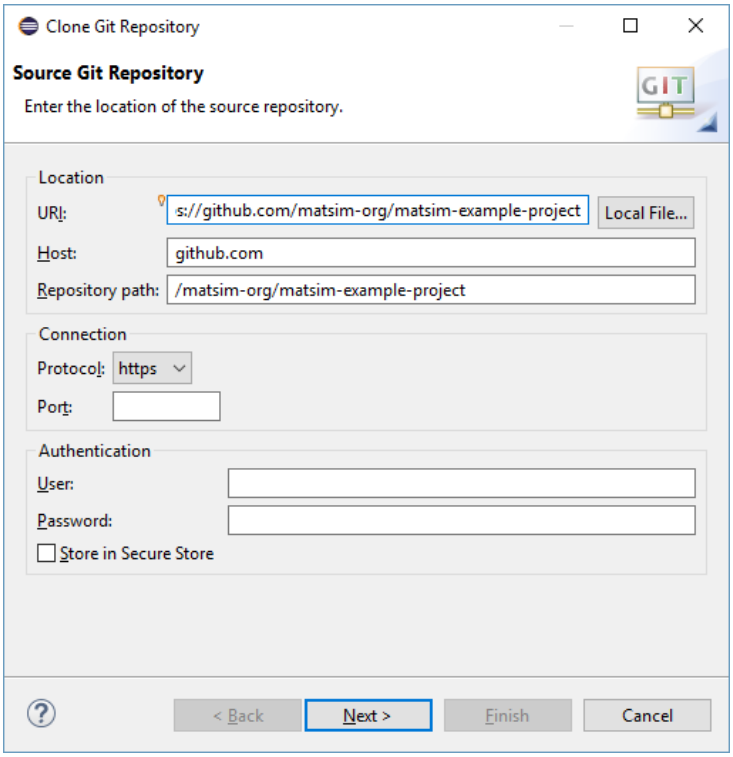
[2] `http://eclipse.org/mars/`
[3] `http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html`
[4] `http://www.pspad.com/cz/`



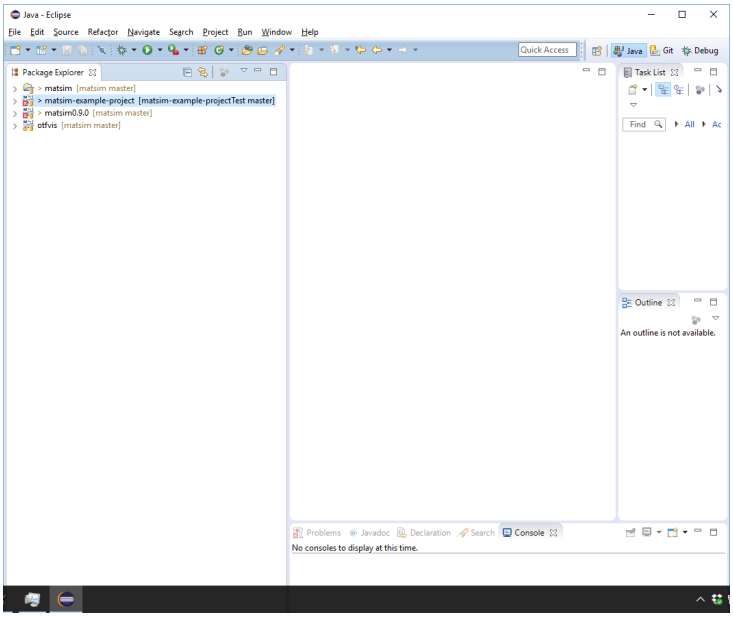
Obrázek 9: Git perspective

- Po přidání se *Git perspective* automaticky zvolí. Nyní vybereme v levém okně *clone git repository*.
- Otevře se okno průvodce. Na první obrazovce vložíme do pole URI adresu Git uložiště matsim-example-projektu `https://github.com/matsim-org/matsim-example-project`, po jejím vložení se nám automaticky doplní některá další pole, ostatní necháme prázdná. Přejdeme na další obrazovku.
- Necháme vybrány větve $SNAPSHOT$ i *master* a přejdeme na další obrazovku.
- Na poslední obrazovce zadáme, kam chceme MATSim uložit a ukončíme průvodce přes $finish$.
- Po stažení souborů se přepneme do Java perspective.
- Zvolíme $File - Import - Maven - existing\ Maven\ projects$.
- Otevře se nám průvodce, zde si přes *browse* zvolíme složku, do které jsme uložili $matsim - example - project$, vybereme pom.xml soubor a dáme $finish$.
- Po stažení všech náležitosti se objeví projekt v *package exploreru*.

Nyní můžeme spustit simulaci nebo vytvořit vlastní rošíření.



Obrázek 10: Git zdroj



Obrázek 11: Git perspective

5.3 Vlastní rozšíření

Předpokladem pro tuto kapitolu je nainstalované vývojové prostředí Eclipse a stažený projekt *matsim – example*.

Spuštění simulace ve vývojovém prostředí provedeme výběrem main souboru, kliknutím pravým tlačítkem na tento soubor a výběru *run as java application*. V případě spuštění přes uživatelské rozhraní je třeba vybrat konfigurační soubor a následně spustit simulaci.

Po dokončení běhu simulace se můžeme podívat do output složky na její výsledek. Abychom mohli vizualizovat výsledek simulace, použijeme program VIA5.4.

Vstupní *main* funkce, je přibližně v tomto tvaru:

```
public static void main(String[] args) {
   String configFile = "cesta ke konfig souboru";
   \\"Nacteni konfigurcniho souburu"
   Config config = ConfigUtils.loadConfig(configFile);
   \\"Nacteni scenare"
   final Scenario scenario = ScenarioUtils.loadScenario(ConfigUtils.loadConfig(
       configFile));
   Controler controler = new Controler(scenario);
   \\spusteni iteraci
   controler.run();
   }
}
```

Výpis 6: main funkce

Konfigurace MATSim simulace je uložena v konfiguračním souboru 4.5.1. Při běhu simulace je načtena do instance $Config$ objektu. Je možné spustit simulaci i bez konfiguračního souboru, nastavením parametrů $Config$ objektu přímo v kódu.

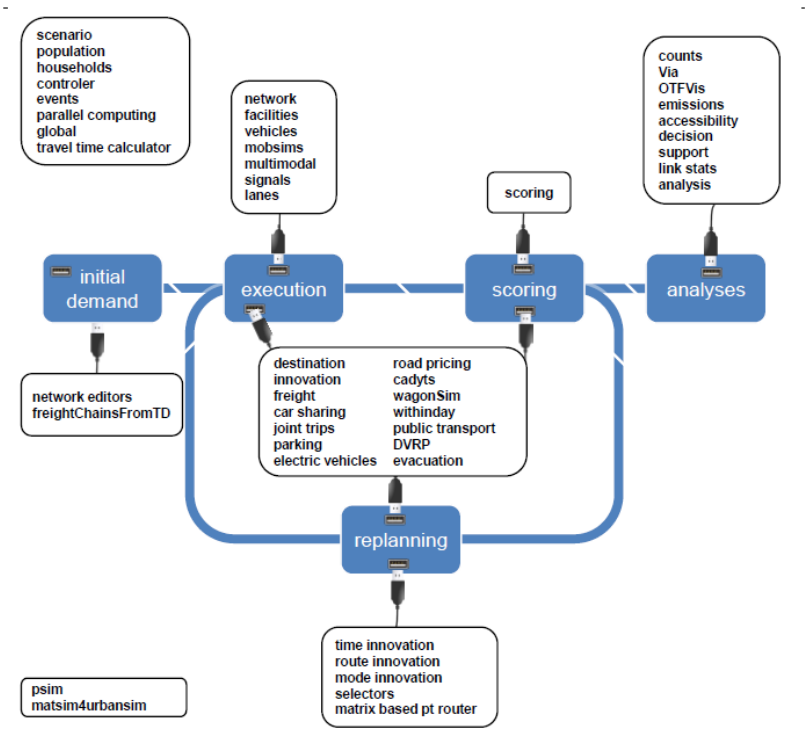
Jak víme, *controler*4.2 řídí iterativní simulaci MATSimu. Každá iterace se skládá ze tří částí - simulace (mobsim), ohodnocení a přeplánování v modulech Obrázek 12, které jsme si představili v sekci 4. Během této simulace mohou být na několika místech zvaných extension points uživatelem zaregistrovány vlastní implementace rozhraní ControllerListener. Tohoto využijeme v dalších sekcích 5.3.4 a 5.3.6 pro implementaci vlastních rozšíření. Ke všem těmto modulům můžeme připojit naše listenery. Moduly mohou být nahrazeny a jsou na sobě nezávislé. Celý stav MATSimu je pouze v *Population*, *Network* objektech a v toku událostí.

Samotné rozšíření o vlastní funkcionalitu provádímě přidáním vlastního modulu, překrytím instalační metody a zaregistrováním vlastních posluchačů7.

```
...
controler.addOverridingModule(new AbstractModule() {
@Override
public void install() {
//Replaning
bindLeastCostPathCalculatorFactory().to(MyLeastPathCostCalculatorFactory.class)
    ;
//WithinDay
this.addEventHandlerBinding().toInstance(travelTime);
this.bind(TravelTime.class).toInstance(travelTime);
this.addMobsimListenerBinding().to(WithinDayMobsimListenerMyNetwork.class);
this.addMobsimListenerBinding().toInstance(travelTime);
}
...
```

Výpis 7: implementace rozšiřovacího bodu

Možnosti rozšíření funkcionality a příslušnosti této funkcionality k modulům jsou zobrazeny na Obrázku 12. Výchozím bodem pro další studium možností rozšíření konkrétní funkcionality je dokumentace. Popisuje jednotlivá API. `http://www.matsim.org/apidocs/core/0.8.1/`



Obrázek 12: MATSim funkcionalita[13]

5.3.1 MATSim struktura

Matsim se skládá z několik balíčků. Zde si uvedeme tři nejdůležitější:

org.matsim.core: Je základním balíčkem, ze kterého je postaveno jádro MATSimu. Obsahuje všechny důležité třídy. Tento balíček by neměl být upravován. V *org.matsim.core* můžeme nalézt důležité třídy jako je controler, mobsim a podobně.

org.matsim.api.core.v1: Veřejné api pro všechny, kteří chtějí používat MATSim a rozšiřovat ho. Obsahuje rozhraní a veřejné metody. Tento balíček by neměl být upravován. *org.matsim.api.core.v*1 obsahuje čtyři důležité části s rozhraními pro události, sítě, populaci a přeplánování.

org.matsim.run: Balíček slouží pro externí spouštění tříd. Každá třída z tohoto balíčku musí obsahovat main funkci pro spuštění z příkazové řádky. V balíčku *org.matsim.run.gui* jsou třídy potřebné pro GUI6 MATSimu.

V MATSimu jsou s rozhraními často i factory třídy. Implementace tříd obsahují v názvech „Impl“. Například $PopulationFactoryImpl$.

Zbývající balíčky je možné nalézt na `http://www.matsim.org/apidocs/core/0.8.1/`.

5.3.2 Implementace MATSim iterace

Zde si probereme jak jsou objektově navrženy tří části MATSim iterace:

Simulace: V simulaci simulujeme plány agenta v síti. V tomto modulu pracujeme s třídou $Population$, reprezentující populaci a třídou $Network$, reprezentující dopravní síť. Výsledkem simulace jsou $Events$ – události. Událostí může být například vstup vozidla na úsek. Každá událost má časové razítko, typ a dodatečné vlastnosti, potřebné pro identifikaci vozidla, úseku. K simulaci toku dopravy by nám měly stačit pouze události.

Ohodnocování: Převod dopravního toku událostí na ohodnocení je pro každého agenta proveden ve třídě implementující $ScoringFunctionFactory$, která vrací ohodnocovací funkce. $ScoringFunctionFactory$ může vracet různé ohodnocovací funkce se specifickými parametry užitečnosti pro různé osoby a jejich plány. Ohodnocovací funkce jsou požadovány v každé iteraci.

Přeplánování: Přeplánování se skládá z několika kroků 4.1. Samotné přeplánování se řídí strategiemi. Strategií může být například přidání nebo odebrání plánu, označení plánu jako vybraného. Strategie jsou implementací $PlanStrategy$ rozhraní. Jsou dvě nejrozšířenější strategie. První je výběr plánu podle specifického algoritmu a jeho označení k provedení. Druhá je náhodný výběr plánu, okopírování, mutace, přidání tohoto plánu do paměti a jeho výběr k provedení. Obě tyto strategie je možné implementovat pomocí pomocných

tříd. *PlanSelector* pro první a *PlanStrategyModule* pro druhou. Nezávisle na tom, co vybraný *PlanStrategy* dělá, MATSim odstraňuje přebývající plány z paměti. Změna trasy při přeplánování je další součásti. K tomuto slouží *TripRouter*. Ten poskytuje metody pro generování cesty mezi lokalitami, podle daného druhu dopravy, času odjezdu a osoby, která přeplánování požaduje. Trasou je v tomto případě sekvence *leg*. *TripRouter* obsahuje mapu (map, dictionary) *RoutingModule* instancí, které jsou svázány s druhy dopravy. *RoutingModule* definuje, jak je cesta tohoto druhu počítána a udává stavy, které generuje při provádění. Pokud bychom potřebovali nový druh dopravy, například jízdu na skateboardu, pak je třeba naimplementovat vlastní *RoutingModule*. *RoutingModule* využívá další dvě rozhraní. *TravelDisutility* a *TravelTime*. *TravelDisutility* udává mezní užitek cestování na úseku v určitou dobu. *TravelTime* udává dobu cesty přes úsek v určitou dobu.

5.3.3 Scenario rozhraní

Osoby (agenti) jsou uloženi v objektové databázi přímo v paměti. Agenti jsou v *Person* objektech společně s jejich plány. Lze k nim přistupovat přes *Population* rozhraní. *Network* rozhraním se dostaneme k dopravní síti, skládající se z článků a uzlů. *Scenario* rozhraní spojuje dohromady výše zmíněné části.

5.3.4 WithinDay plánování

Nyní si rozebereme problém rozšiřování funkcionality na konkrétním příkladu. Ve skutečném světě nastávají během dne různé neplánované, nahodilé události, jako jsou například dopravní nehody, přírodní katastrofy a podobně. MATSim vypočítává simulaci v iterativním procesu. Agenti se učí z předchozích zkušeností a podobných situací, které prožili v minulosti. V případě využití neočekávaných událostí dojde k chybným výsledkům a neracionálnímu chování. Agent nemůže vědět, že dojde k neočekávané události. Abychom tomuto předešli, využijeme techniku $within - day - replaning$, která simuluje pouze jednu iteraci a dokáže upravovat trasu během dne nebo když je agent na cestě. Toto se netýká všech agentů. Agenti implementující rozhraní pro veřejnou dopravu *TransitDriverAgentImpl* a *TransitAgent* nepodporují přeplánování.

Potřebujeme dosáhnout úpravy při běhu mobsimu a načítání sítě. Toho můžeme dosáhnout implementací MobsimBeforeSimStepListener.

```java
package org.matsim.withindayreplaning;

import org.matsim.core.mobsim.framework.events.MobsimBeforeSimStepEvent;
import org.matsim.core.mobsim.framework.listeners.MobsimBeforeSimStepListener;

public class WithinDayMobsimListener implements MobsimBeforeSimStepListener {

@Override
public void notifyMobsimBeforeSimStep(MobsimBeforeSimStepEvent e) {
// TODO Auto-generated method stub

}

}
```

Výpis 8: implementace MobsimBeforeSimStepListener posluchače

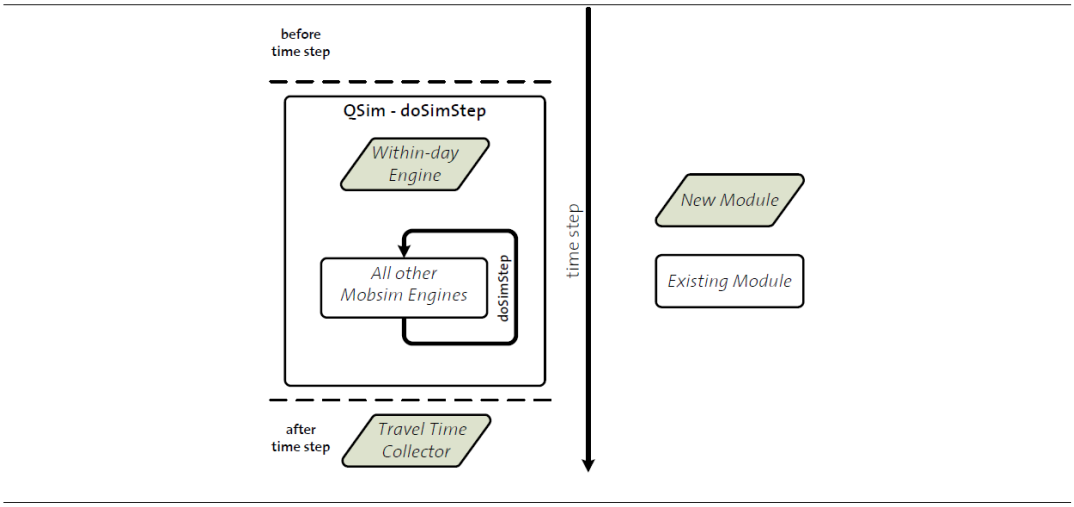
Pro každého agenta provedeme přeplánování. Implementace přeplánování je realizována *MobsimEnginem* připojeným do QSimu, ten je volán při každém časovém kroku simulace. Dojde tedy ke změně plánu agenta. *MobsimEnginu* můžeme připojit několik a každý může upravovat specifické agenty. Výběr agentů provádí tzv. *AgentSelector*. Další částí je *Replaner*, ten není zodpovědný za výběr agentů pro přeplánování, není zodpovědný za zjišťování, zda má být agent přeplánován. *AgentSelector* se stará o to, zda jsou k přeplánování trasy vybírání pouze agenti co provádí *leg* - tedy jsou na cestě. *AgentSelector* nesmí přeplánovávat plán a měnit trasu, o to se stará *Replaner*. Výsledkem tohoto je možné provádět přeplánování paralelně a výrazně urychlit simulaci.

Protože je výběr agentů *selektorem* rychlý, je prováděn sekvenčně. Paralelním výběrem by nedošlo k žádnému citelnému zrychlení.

Přeplánování umožňuje změnu jakékoliv položky agentova denního plánu. Všechny cesty a aktivity mohou být upraveny. Pouze některé operace nemohou být změněny, pokud už aktivita začala. Možné změny:

- Aktuální cesta (trasa, cíl).
- Budoucí cesta (přidání, odebrání, cesta, počátek, cíl).
- Aktuální aktivita (konečná doba).
- Budoucí aktivita (přidání, odebrání, umístění, typ, počáteční a koncový čas).

Simulace QSimu je rozdělena do tří fází. Fáze před vykonáním kroku, fáze vykonání kroku a fáze po vykonání kroku. Během druhé fáze všechny registrované *MobsimEngines* simulují



Obrázek 13: Fáze QSimu[13]

aktuální časový krok. *WithinDayEngine* je vždy první engine vyhodnocující své kroky během druhé fáze, což nám zajistí, že žádný z agentů nezmění svůj stav během předchozího časového běhu, tedy výběr probíhá nad aktuálnímy daty.

5.3.5 Implementace

Implementujeme *MobsimBeforeSimStepListener*. Jak jsme si řekli výše, je potřeba implementovat dvě operace. Výběr agentů a přeplánování. Naší metodou pro výběr je *getAgentsToReplan*.

```
private static List<MobsimAgent> getAgentsToReplan(Netsim mobsim) {
List<MobsimAgent> set = new ArrayList<MobsimAgent>();

if(Math.floor(mobsim.getSimTimer().getTimeOfDay()) <= 5 * 60 * 60 ) {
   return set;
}

for (NetsimLink link:mobsim.getNetsimNetwork().getNetsimLinks().values()){
   for (MobsimVehicle vehicle : link.getAllNonParkedVehicles()) {
      MobsimDriverAgent agent=vehicle.getDriver();
      if ( true ) {
      set.add(agent);
      }
   }
}

return set;
}
```

Výpis 9: implementace getAgentsToReplan

V podmínce *if* je omezení doby, kdy provádíme výběr. Pokud je čas dřívější než pět hodin ráno, není prováděn žádný výběr agentů k přeplánování. V následujícím cyklu jsou procházeny všechny články a vozidla na nich, jsou vybráni agenti řídící vozidlo. Funkce vrací množinu agentů vhodných pro přeplánování.

Metoda provádějící přeplánování se nazývá *doReplanning*. V metodě využíváme pomocné metody ze třídy *WithinDayUtils*.

5.3.6 Vlastní hodnocení cesty

Pokud chceme ovlivnit výběr cesty, je možné si připravit vlastní algoritmus, vyhledávající cestu v síti. Opět si přidáme vlastní modul pro výpočet nejkratší cesty v grafu. Modul implementuje rozhraní *LeastCostPathCalculatorFactory*, vytvářející konkrétní objekty počítající ceny cest v síti. Konkrétní implementace výpočtu nejkratší cesty v síti je *LeastCostPathCalculator*. Jedná se vlastně o počítání nejkratší cesty v grafu. V příkladu je implementace Dijksrtova algoritmu5.3.7 pro výpočet nejkratší cesty v grafu. Pro implementaci do MATSimu musíme nejdříve vytvořit třídu dědící z *LeastCostPathCalculatorFactory* - jedná se o

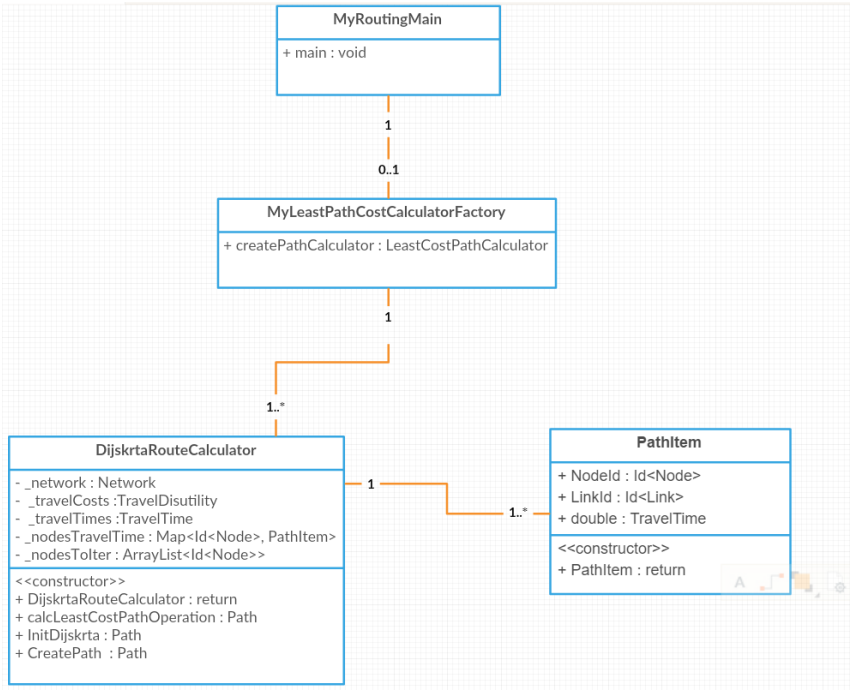
MyLeastPathCostCalculatorFactory, ta obsahuje motodu vracející náš *DijskrtaRouteCalculator* implementující rozhraní *LeastCostPathCalculator* a obstarávající samotný výpočet nejkratší cesty v grafu dle Dijskrtova algoritmu. Na Obrázku 14 je UML diagram tříd implementace.

5.3.7 Dijkstrův algoritmus

Dijkstrův algoritmus je nejznámější algoritmus pro nalezení nejkratších cest v grafu z počátečního vrcholu. Vstupem je neorientovaný graf $G = \langle V, E \rangle$. V je množina vrcholů a E je množina hran. Hranové ohodnocení $w : E \to \mathbb{R}^+$. Mějme počáteční vrchol $s \in V$. Výstupem algoritmu je hodnota d(v) pro každý $v \in V$. d(v) je hodnota nejkratší cesty z s do v. Dále máme množiny vrcholů A *a* $N \subseteq V$, funkci $d : V \to \mathbb{R}_0^+$ a číslo $m \in \mathbb{R}_0^+$.

Na počátku obsahuje množina A všechny vrcholy V. Počáteční vrchol d(s) = 0 a všechny vrcholy $v \in V - \{s\}$ platí $d(v) = \infty$.

1. Pokud neexistuje $v \in A$, kde $d(v) \neq \infty$ skončíme.
2. $m = min\{d(v)|v \in A\}$, $N = \{v \in A|d(v) = m\}$ a množina A = A-N
3. Nyní pro každý vrchol $v \in N$ a $u \in A$ takové, že $\{v, u\} \in E$ provedeme výpočet d(v) + w(v,u) < d(u) pak je d(u) = d(v) + w(v,u) a pokračujeme prvním bodem.



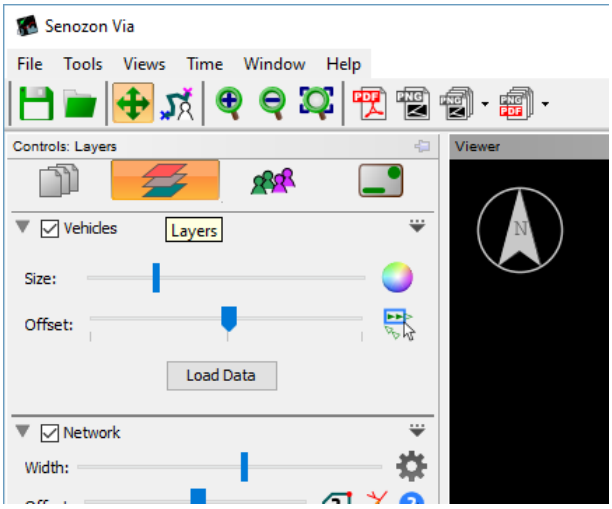
Obrázek 14: Diagram tříd Dijkstrova algoritmu

5.4 Via - nástroj pro analýzu a simulaci

Výsledkem běhu MATSimu jsou události. Pokud si zobrazíme výsledek v textovém editoru, je jeho analýza velmi náročná. Může se jednat o soubor velikosti několika set MB s mnohaset tisíci řádky, tedy jeho analýza v textovém editoru je nemožná.

Via [5] je analytický nástroj pro vizualizaci a analýzu výsledku simulace MATSimu 4.6.4. Dokáže vizualizovat MATSim síť a vygenerované události. Zobrazit vznikající zácpy na úsecích, omezovat zobrazovaná data pouze na určité skupiny osob. Před použitím a spuštěním je třeba se zaregistrovat, stáhnout si volnou licenci pro nekomerční použití omezenou na 500 agentů. Pro zobrazení výsledku MATSim analýzy je potřeba provést několik kroků:

- Přidání sítě. Zvolíme $File-addData$ - vybereme $network.xml$ soubor použitý při simulaci.
- Výsledné události z běhu simulace. Zvolíme $File-addData$ - vybereme soubor ze složky $output-iters-it.10$ - tedy poslední iteraci.
- Přidání vrstvy sítě. Zvolíme $File-AddLayer$ - zde vybereme ve stromě $Network$ a vpravo nahranou síť. Síť se nám zobrazí v prohlížeči.
- Přidání událostí. Zvolíme $File-AddLayer$ - zde vybereme ve stromě $Vehicles$ a vpravo nahrané události.
- Pro zobrazení událostí se přepneme v hlavním okně na druhou záložku a načteme události tlačítkem $LoadData$.



Obrázek 15: VIA

- Ve spodní části okna se nám zobrazí časová osa společně s posuvníkem pro nastavení rychlosti přehrávání událostí.

[5] https://via.senozon.com/

6 Testování MATSimu

V této kapitole se budéme zabývat testováním MATSimu. Vyzkoušíme různé pokusy nad dvěma scénáři. Otestujeme naši implementaci WithinDay5.3.4 přeplánování, testovat také budeme vliv počtu iterací na kvalitu výsledku a otestujeme i škálování na vícejádrových procesorech. První sekce popisuje testovací systém a testovací data. Další sekce uvádí výsledky testů nad testovacími scénáři.

6.1 Systém pro testování

Základem testů byla vlastní funkcionalita naimplementovaná v předchozích sekcích 5. Testy byly spouštěny z vývojového prostředí Eclipse.

Testovací sestava, na které byly testy prováděny se skládá:

- Operační systém Microsoft Windows 10.
- Vývojové prostředí Eclipse verze MARS `http://eclipse.org/mars/`.
- Počítač osazen processorem Intel i7 4790k s 8 GB operační paměti.

Scénáře, na kterých byly prováděny testy:

- CEMDAP-MATSim-Cadyts scénář města Berlín.
- Malá testovací síť.

6.1.1 Berlín

Berlín je hlavním městem Německa s více než třemi miliony obyvatel. Abychom mohli modelovat chování populace ve městě, potřebujeme data o jejích cestovních návycích a místech aktivit. Uvedené informace ale v mnoha případech podléhají utajení, proto není jednoduché podobný scénář sestavit.

Testovací prostředí Berlín (CEMDAP-MATSim-Cadyts scenario)[23] je postaveno na volně dostupných datech, která již byla anonymizována pro jejich původní využití. Zakladním zdrojem jsou data ze sociálního pojištění, obsahující informace o umístění domova a zaměstnání pracovníků. Abychom dostali detailnější informace, je potřeba provést dva kroky. Prvním krokem je zvětšení rozlišení dat. Druhým krokem je modelování ostatních aktivit, ne jen domov a práce.

První krok byl proveden pomocí postupu [7] v softwaru Cadyts. Cadyts během iterativní simulace MATSimu odhadne pravděpodobná propojení dle dodaných dat. Tedy dostaneme pravděpodobné výskyty aktivit v rámci denního plánu agenta tak, aby odpovídaly skutečnosti.

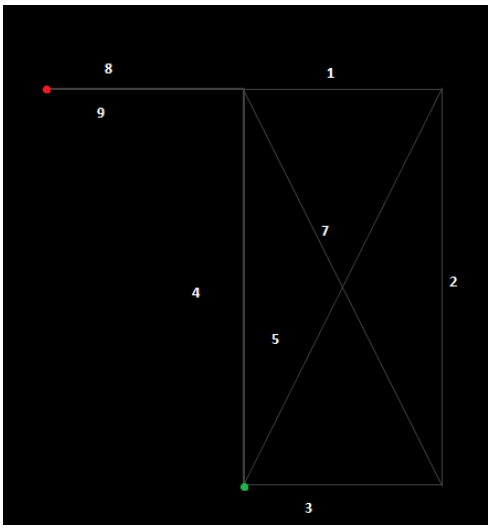
Druhý krok - doplnění dalších aktivit pro jednotlivé agenty. K tomuto byl použit proces CEMDAP [3] na základě demografie.

Celkový proces vytvoření scénáře se skládá z dat o domovech a pracovních místech. Ty jsou náhodně přiřazeny každému agentovi. K doplnění denních plánů o další aktivity je použit CEMDAP. Nakonec je použit Cadyts pro zkalibrování plánů agentů tak, aby co nejvíce odpovídaly dopravní situaci v Berlíně dle studie [1].

Výsledkem jsou dva scénáře s 1% a 10% populací. Jedná se tedy o 16 000 nebo 160 000 agentů. Síť scénáře obsahuje více než 10 000 uzlů spojených 30 000 úseky. Aktivity ve scénaři jsou "home", "work", "leis", "shop"a "other".

6.1.2 Malá testovací síť

Tuto malou testovací síť jsme vytvořili pro otestování přeplánovní. Síť je velmi jednoduchá a přehledná, tedy je možné i pouhým pohledem zjistit, zda se agenti chovají dle očekávání. Tvar sítě je vidět na Obrázku 16. V této síti mají agenti za úkol dostat se z počátečního červeného bodu představujícího domov, do práce v zeleném bodě. Následně se opět vrátit domů. Tedy dostat se z vrcholu 5 do vrcholu 4. Scénář obsahuje aktivity "home" a "work". Nejkratší cesta vede přes hrany 8 - 7 - 3. Ve scénáři je nastavena rychlost na všech úsecích 27.78 $\mathrm{m\,s^{-1}}$.



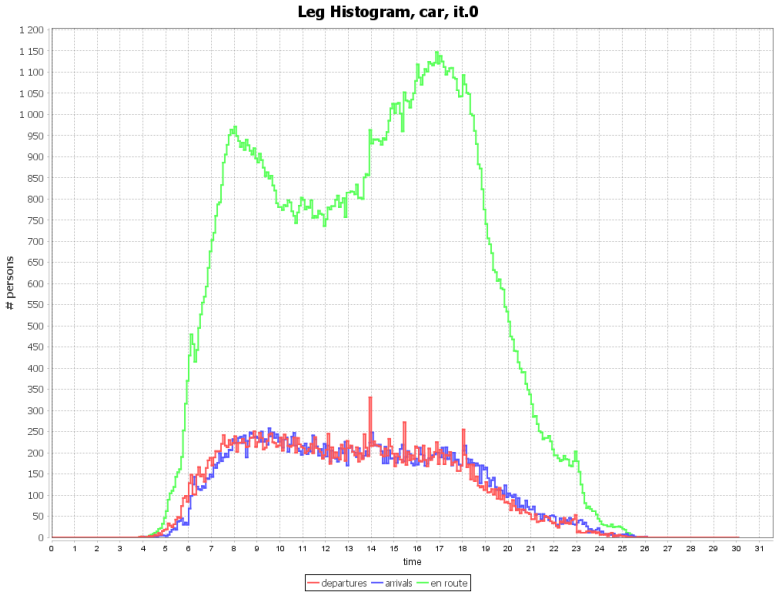
Obrázek 16: Testovací síť pro WithinDay přeplánování

6.2 Výsledky testů Berlín

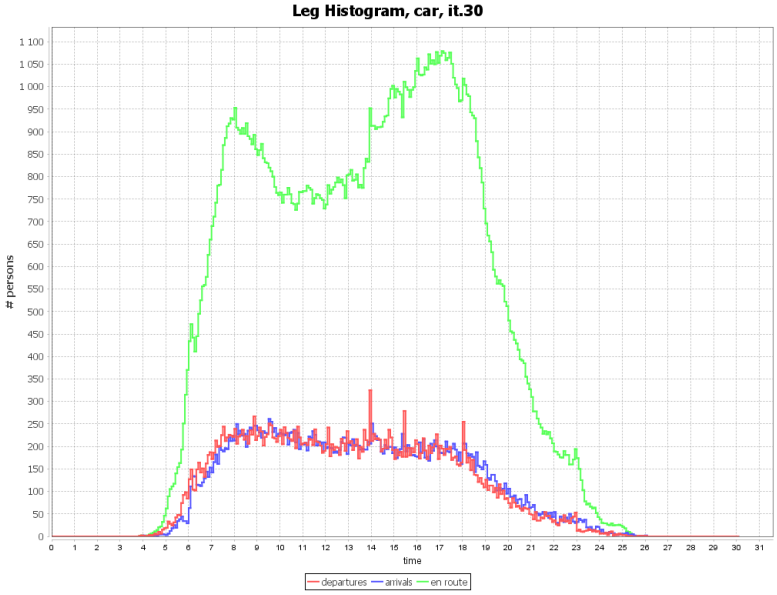
V testech nás zajímá dosažené skóre simulace. Tedy jakého ohodnocení simulace dosáhla. Naším cílem je dosahnout co nejvyššího skóre. Přirozeně je lépe hodnocena kratší cesta agenta s kratší dobou jízdy než delší. Nicméně nesmí to být na úkor ostatních agentů. Snažíme se dosáhnout optimálního stavu. Skóre by mělo s každou iterací stoupat. Data byla použita pro scénář Berlín s 1% populací [23]. Simulace pro 30 iterací s 1% populací trvala přibližně 16 minut. Výsledná velikost events souboru je 139MB. S 10% populací se 160 000 agenty trvaly testy neúměrně dlouho. V tomto případě výpočet pro pouhé 2 iterace trval 25 minut.

Na obrázcích jsou grafy znázorňující počty agentů v průběhu dne. Červenou barvou jsou znázornění agenti připravující se k odjezdu, modrou barvou agenti, kteří dojeli do cíle a zelenou barvou agenti na cestě.

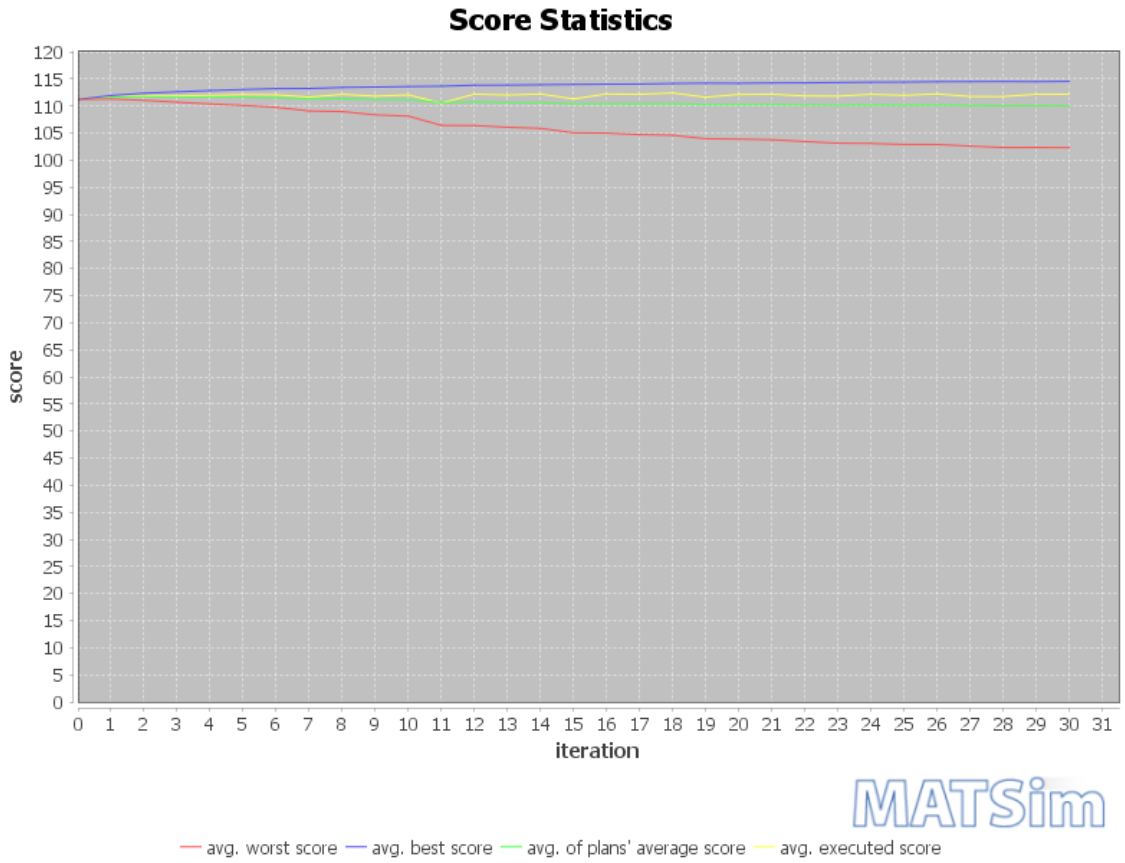
Na prvním Obrázku 17 je graf znázorňující první iteraci, na druhém Obrázku 18 poslední třicátou iteraci.



Obrázek 17: Berlín, počty agentů na cestě, 1. iterace



Obrázek 18: Berlín, počet agentů na ceste, 30. iterace



Obrázek 19: Berlín, vývoj skóre

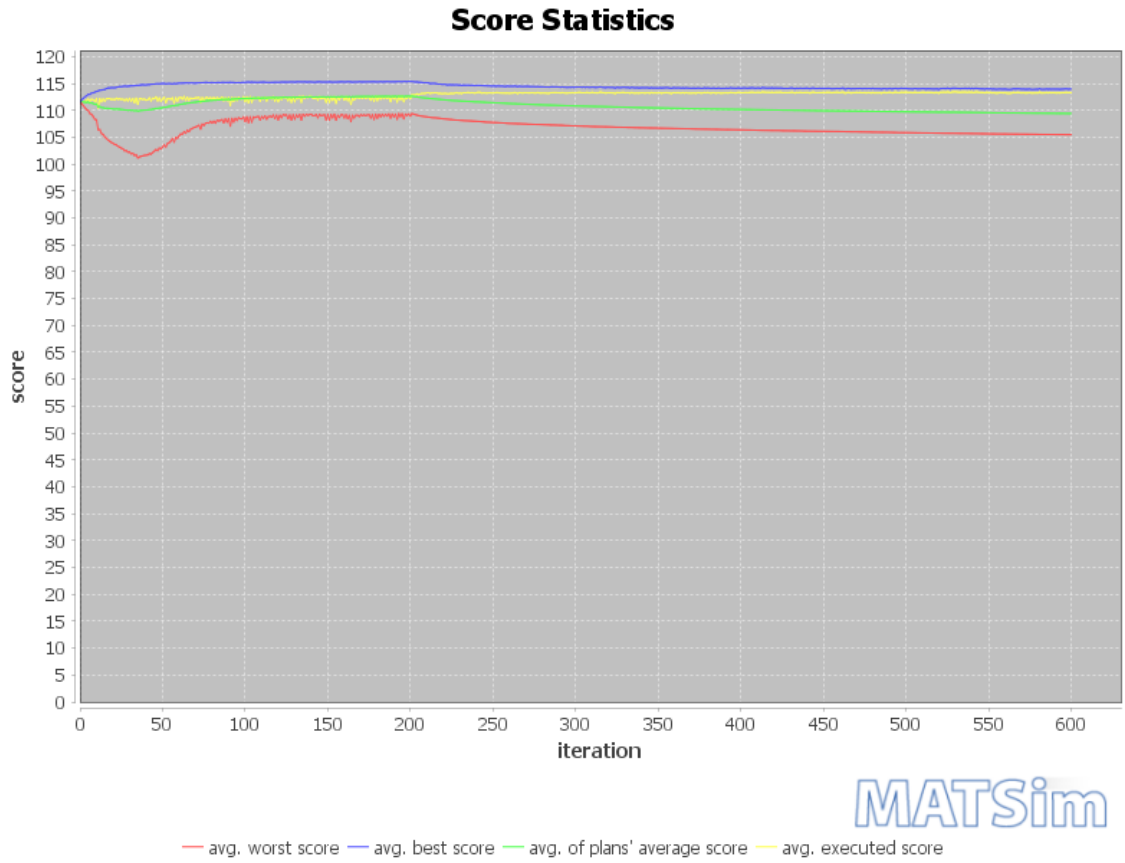
Po první iteraci bylo skóre simulace 111.21. Průměrná doba cesty agenta je 00:23:06 min a celková ujetá vzdálenost je 13257.15 m. Po poslední iteraci je skóre simulace 114.56, průměrná doba cesty je 00:22:22 min a ujetá vzdálenost 13251.71 m.

Při porovnání grafů je vidět, že se mírně snížily vrcholy v 7 a 17 hodin. Při první iteraci dosahoval vrchol v 7:00 více jak 950 agentů a v 17:00 1150 agentů, po poslední iteraci dosahoval v 7:00 přibližně 950 agentů a v 17:00 asi 1070 agentů. Doprava se rozprostřela před a za tyto vrcholy.

Vývoj skóre je vidět na Obrázku 19. Během iterací rostlo skóre od 111.21 v první iteraci až po 114.56 v poslední. Postupně se během iterací zvyšoval i rozdíl mezi nejlepším a nejhorším průměrným skóre. V další sekci se budeme zabývat vývojem skóre podle počtu iterací.

6.3 Vliv počtu iterací na skóre

Dalším zajímavým pokusem bylo zjištění vlivu počtu iterací na výsledné skóre. Předpokladem je zvyšující se skóre až do optimálního bodu, ve kterém již nebude docházet k žádnému zlepšní. Pro test byl zvolen počet šestiset iterací.



Obrázek 20: Vývoj skóre, 600 iterací

Na Obrázku 20 se v grafu potvrdila rostoucí tendence zvyšování skóre. Od první iterace se skóre 111.29 postupně zvyšovalo až po šedesátoupátou iteraci se skóre 115.12. Poté dochází ke stagnaci a v dalších iteracích již k výraznějšímu nárůstu nedojde. Do dvousté iterace je nalezeno řešení se skóre 115.4, které je nejvyšší nalezenou hodnotou. Od dvousté iterace dochází spíše k propadu skóre a dle výsledku pokusu nemá příliš velkou cenu dále pokračovat.

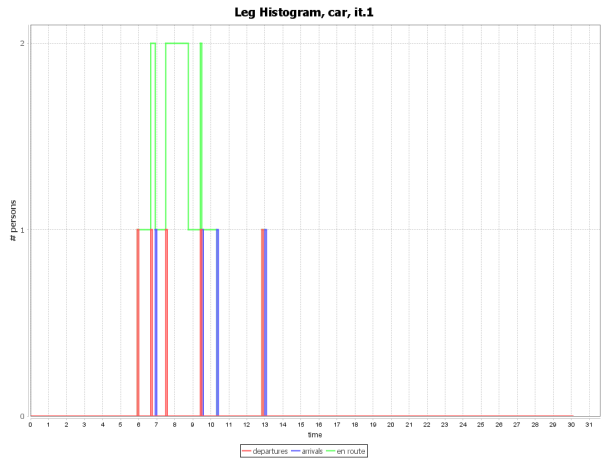
Optimální počet iterací je závislý na velikosti scénáře. Optimální počet pro Berlín se jeví 200 iterací. Pro malou síť použitou ve WithniDay přeplánování stačí 30 iterací.

6.3.1 WithinDay přeplánování

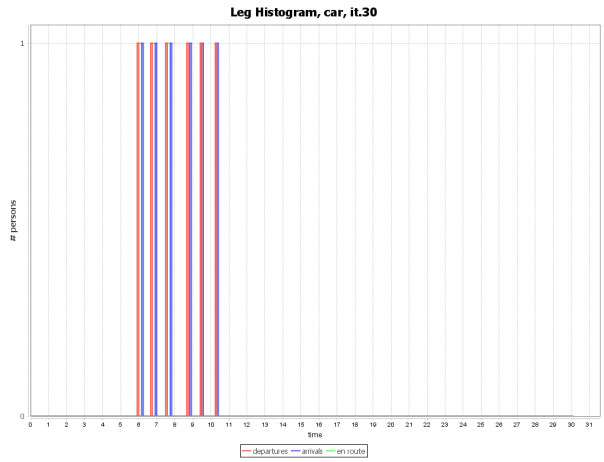
*WithinDay*5.3.4 přeplánování bylo zkoušeno na malé testovací síti 6.1.2.

Pomocí *WithinDay* plánování je mezi 5. a 10. hodinou ranní omezena rychlost na úseku 7 na rychlost 1. Očekávaným výsledkem by mělo být přeplánování cesty tak, aby byly použity úseky 1 a 2 místo úseku 7. První agent se vydá na cestu v 6 hodin, druhý v 6:40 a třetí v 7:30. Možnost přeplánování je agentům umožněna od 6:06 do 7:00 hodin. Zajímá nás po kolika iteracích naleznou všichni agenti optimální cestu.

Výsledek první simulace je vidět na Obrázku21. První agent cestuje přes pomalý úsek 7, agent nemá umožněno přeplánování své cesty. Agentovi trvá cesta z domova do práce hodinu a půl. 2. agent vyráží na cestu v době, kdy je umožněno přeplánování, čehož využije a upraví trasu průjezdem přes úseky 1 a 2. Třetí agent opět nemá umožněno přeplánování, tedy opět cestuje přes úsek 7. Průměrná doba cesty je 51:25 minut.



Obrázek 21: Cestovní histogram, první iterace

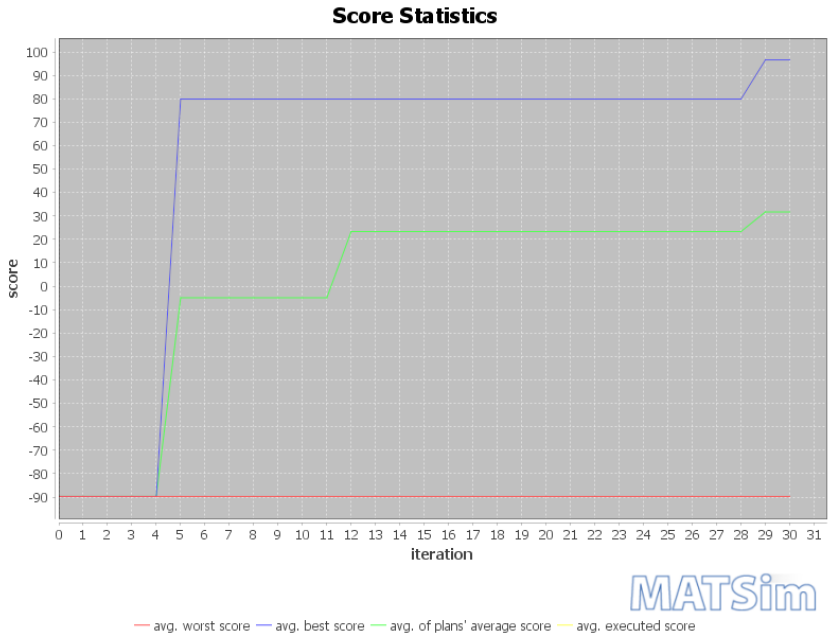


Obrázek 22: Cestovní histogram, třicátá iterace

Ve třicátém průchodu viz. Obrázek22 již všichni agenti cestují přes úseky 1 a 2. Průměrná doba na cestě se snížíla na 12:00 minut.

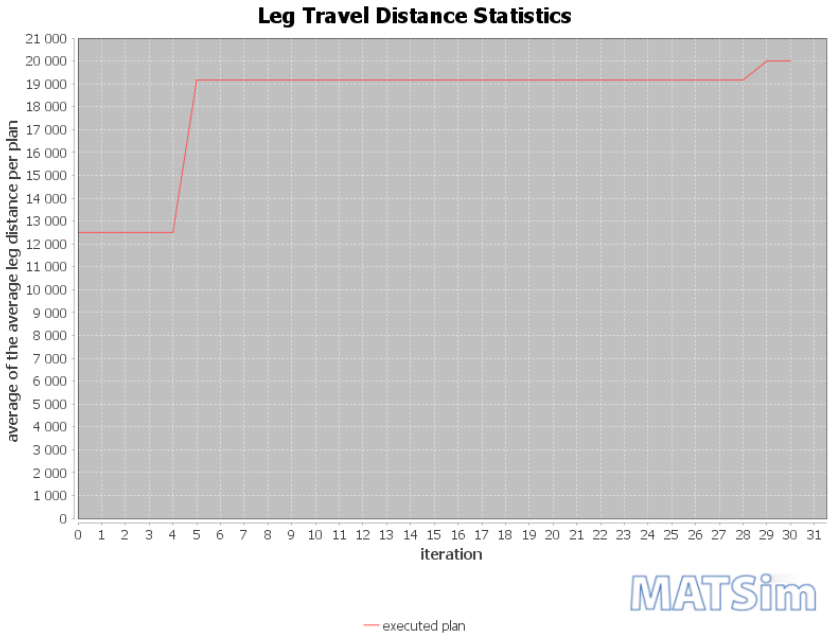
Z grafu je vidět, že se první agent adaptoval na kratší cestu po čtyřech iteracích. Třetí agent až po 28 iteracích.

Další dva obrázky ukazují vývoj skóre a absolvovanou vzdálenost.



Obrázek 23: Vývoj skóre

Na obrázku24 se ujetá vzdálenost zvětšila z počátečních 12500 na konečných 20000.



Obrázek 24: Vývoj průměrné ujeté vzdálenosti

Jak je vidět z předchozích výsledků, kvalita řešení (výška skóre) nezáleží pouze na ujeté vzdálenosti. Vzdálenost se v našem případě zvýšila, přesto se zlepšilo výsledné skóre díky snížení doby na cestě z 51:25 na 12:00 minut.

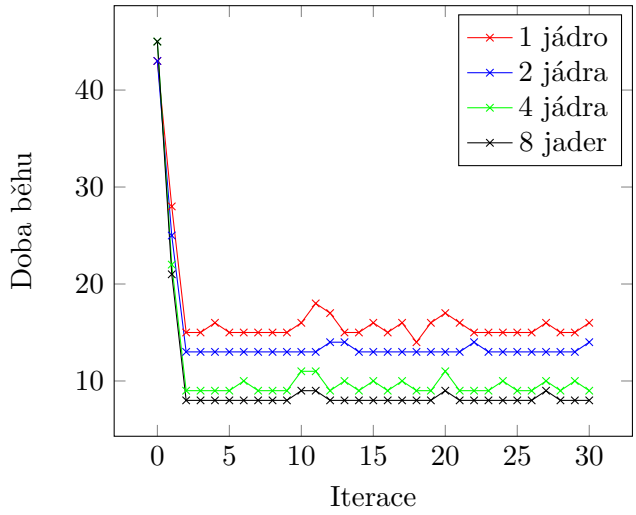
6.4 Paralelizace simulace MATSimu

Dalším zajímavým pokusem bylo zjištění škálování simulace MATSimu při paralelním běhu. MATSim umožňuje paralelní běh ve více vláknech. Je doporučeno začít s počtem dostupných jader procesoru. Nastavení se provádí v konfiguračním souboru. V konfiguračním4.5.1 souboru byl nastaven v globálním modulu parametr *numberOfThreads* na požadovanou hodnotu. Parametr *numberOfThreads* je nastaven i pro qsim modul.

6.4.1 Výsledky

Test byl proveden pro 1, 2, 4 a 8 vláken. Jako procesor byl použit Intel Core i7 4790 se čtyřmi fyzickými jádry a schopností Hyper-threadingu, tedy paralelizace fyzického jádra na dvě. Díky tomu je v systému 8 jader. Níže jsou uvedeny jednotlivé výsledky pro běh 30 iterací nad 1% populace Berlína.

Dle výsledků je vidět zlepšení při zapojení paralelizace výpočtu. V tabulce1 je souhrn výsledků pro jednotlivé pokusy.



Počet jader	Nejkratší běh	Nejdelší běh
1	16	43
2	14	43
4	10	45
8	8	45

Tabulka 1: Paralelní běh

Pro jedno vlákno trvala iterace v průměru 16 vteřin, pro dvě vlákna v průměru 14 vteřin, pro čtyři vlákna v průměru 9-10 vteřin. Pro 8 vláken již neproběhlo výraznější zlepšení. V průměru trvala jedna iterace 8 vteřin.

Jak je vidět, paralelizace poměrně citelně zrychlila běh.

6.4.2 Vliv Hyper-threading

Byly provedeny i testy bez aktivovaného Hyper-threadingu aby se potvrdil nebo vyvrátil jeho vliv na rychlost. Hyper-threading je technologie od společnosti Intel. Jde o vytvoření dvou virtuálních processorů z jednoho fyzického. To se děje aktivací dvou řídících jednotek v procesoru.

Test byl proveden pro dvou a čtyř jádrové nastavení. Ani v jednom případě se nezměnila doba výpočtu. Pro dvě jádra byla průměrná doba 14 vteřin. Pro čtyři jádra 10 vteřin. Nepotvrdil se negativní účinek technologie Hyper-threading na dobu běhu.

6.4.3 Vliv paralelizace na výsledné skóre

Dále nás zajímalo, zda se paralelizace negativně nepodepisuje na kvalitě získaného výsledku, hodnotě skóre.

Počet jader	Počáteční skóre	Výsledné skóre
1	111.209	114.617
2	111.297	114.528
4	111.297	114.528
8	111.297	114.528

Z výsledků testů se nepotvrdil negativní vliv paralelizace na výsledné skóre.

7 Závěr

V prácí jsme se seznámili s pojmy jako je model a simulace. Tyto pojmy jsme zasadili do kontextu dopravních simulací. Uvedli jsme si, jak vůbec dopravní situaci modelovat a reprezentovat v ní pohyb pomocí agentů. U agentů jsme se seznámili s jejich vlastnostmi a vlastnostmi prostředí, ve kterém operují, dále s pojmem softwarový agent a multiagentní systém, který je velmi podstatný pro modelování dopravy. Lehce jsme se dotkli i umělé inteligence, protože multiagentní systémy vykazují její určité znaky.

Po představení teorie jsme se konečně dostali k MATSimu, tedy multiagentímu dopravnímu simulátoru. Řekli jsme si, jak simulace v MATsimu funguje, že se jedná o iterativní proces. Dále jsme probrali různé detaily implementace. Velmi podstatnou sekcí bylo, jak MATSim ohodnocuje simulaci. V předposlední sekci jsme se seznámili s rozšiřováním MATSimu a uvedením potřebných nástrojů k jeho zprovoznění. Na dvou příkladech jsme si ukázali jak konkrétně MATSim rozšířit o námi zamýšlenou funkcionalitu a nakonec jak si zobrazit výsledky simulace MATSimu v analytickém nástroji VIA. Poslední kapitola se zabývala testováním MATSimu, představili jsme si dva scénáře, provedli testy chování MATSimu a zjistili, zda je možné zrychlit běh MATSimu paralelizací.

Při studiu se ukázal MATSim i bez rozšiřování jako velmi komplexní nástroj pro simulaci dopravy s velkým potenciálem pro reálné využití. Díky možnosti rozšiřování je možné MATSim dále upravit pro specifické úkoly. Dalším zajímavým tématem, navazujícím na tuto práci, by bylo vytvoření reálného scénáře českého města pro další studium.

Literatura

[1] F. Ließke R. Wittwer Ahrens, G.-A. and S. Hubrich. Endbericht zur verkehrserhebung 'mobiität in städten – srv 2008' und auswertungen zum srv-städtepegel. 2009.

[2] Richard Arnott, André de Palma, and Robin Lindsey. Economics of a bottleneck. *Journal of Urban Economics*, 27(1):111 – 130, 1990.

[3] Guo J. Srinivasan S. Bhat, C. and A Sivakumar. Cemdap users's manual. 3, 2008.

[4] Peter Braun and Wilhelm Rossak. *Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[5] Ennio Cascetta. Transportation systems engineering: theory and methods, 2013.

[6] Axhausen Kay W Charypar David, Balmer Michael.

[7] Gunnar Flötteröd, Michel Bierlaire, and Kai Nagel. Bayesian demand calibration for dynamic traffic simulations. *Transportation Science*, 45(4):541–561, 2011.

[8] Mogens Fosgerau and Leonid Engelson. The value of travel time variance. *Transportation Research. Part B: Methodological*, 45(1):1–8, 2011.

[9] Stan Franklin and Art Graesser. *Is It an agent, or just a program?: A taxonomy for autonomous agents*, pages 21–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[10] Denos C. Gazis. The origins of traffic theory. *Oper. Res.*, 50(1):69–77, February 2002.

[11] Karsten Hager, Jürgen Rauh, and Wolfgang Rid. Agent-based modeling of traffic behavior in growing metropolitan areas. *Transportation Research Procedia*, 10:306 – 315, 2015.

[12] M. Hilliges.

[13] Nagel K. Horni, A. and K.W. Axhausen. *The Multi-Agent Transport Simulation MATSim.* Ubiquity Press., 2016.

[14] Joel L Horowitz. Discrete choice analysis: Theory and application to travel demand, by m. ben-akiva and s.r. lerman, mit press, 1985: Transportation science, 1986. Type: Review.

[15] Michael N. Huhns and Larry M. Stephens. Multiagent systems. chapter Multiagent Systems and Societies of Agents, pages 79–120. MIT Press, Cambridge, MA, USA, 1999.

[16] Fan Linna and Liu Jun. A free-roaming mobile agent security protocol against colluded truncation attack. In *2010 2nd International Conference on Education Technology and Computer*, volume 5, pages V5–261–V5–265, June 2010.

[17] Henry X. Liu, Xiaozheng He, and Bingsheng He. Method of successive weighted averages (mswa) and self-regulated averaging schemes for solving stochastic user equilibrium problem. *Networks and Spatial Economics*, 9(4):485–503, 2009.

[18] Matsim.org. Matsim loop, 2017. [Online; accessed April 17, 2017].

[19] Richard Murch and Tony Johnson. *Intelligent Software Agents*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[20] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[21] V. I. Shvetsov. Mathematical modeling of traffic flows. *Automation and Remote Control*, 64(11):1651–1689, 2003.

[22] William S. Vickrey. Congestion theory and transport investment. *The American Economic Review*, 59(2):251–260, 1969.

[23] vsp.tu berlin. vsp.tu-berlin scenario.

[24] Wikipedie. Náklady obětované příležitosti — wikipedie: Otevřená encyklopedie, 2013. [Online; navštíveno 19. 04. 2017].

[25] Wikipedie. Mezní užitek — wikipedie: Otevřená encyklopedie, 2016. [Online; navštíveno 18. 04. 2017].

[26] Wikipedie. Utm — wikipedie: Otevřená encyklopedie, 2016. [Online; navštíveno 18. 04. 2017].

[27] Wikipedie. Sférická soustava souřadnic — wikipedie: Otevřená encyklopedie, 2017. [Online; navštíveno 18. 04. 2017].

[28] Michael Wooldridge. *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition, 2009.

A Příloha na CD/DVD

Obsah CD/DVD:

- Soubory pro běh scénářů.
- Zdrojové soubory rozšíření MATSimu.
- Readme soubor.